C# Programming Language

- ان لغة السى شارب تتبع مجموعة لغات Net Frame Work. وهذه البيئه
- تحتوى على عدة لغات لكن كل منهم له الكود الخاص به وتوجد بها لغه وسيطيه Microsoft Intermediate Language (MSIL) وبالتالى هى لغه لا تعتمد على الاله فهى تعمل على اى جهاز

- ان C# تضم مميزات VB - C++ بما لها من سهولة التعامل فى الواجهات الرسوميه بالاضافه الى قوة البرمجه .

- ان الفارق الجوهرى بين C# و C++ هو كالتالى :

C#	***C++***
هى لغه مبنيه بالاساس على مفهوم البرمجه كائنية التوجه او Object Oriented	هى لغة تستطيع بها ان تستخدم مفهوم البرمجه كائنية التوجه

الهيكل التنظيمى لبرنامج #C :



نأتى لتفسير التكوين الرئيسى لبرنامج #C

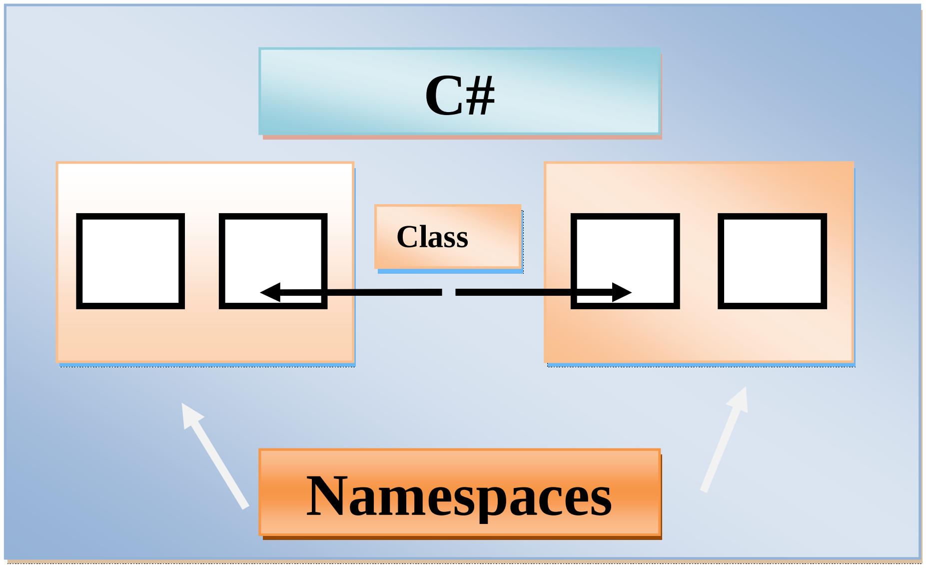
<u>Using</u> هى كلمه محجوزه داخل اللغه وتستخدم للنداء على اسم معين ليصبح

جزء من البرنامج

<u>Namespace</u> ويعرف مجازا بفضاء الاسماء وهو عباره غن مجلد يحتوى

بداخله على <u>Classes</u> جاهزه للعمل عليها ولها اسماء محدده

ونستطيع انه نمثله بالشكل التالى :



Classes هى وحدات محجوزه داخل اللغة وتحتوى على جزئين رئيسين:

1- Data Member وهى المتغيرات والتى تحتوى على بيانات

2- Function Member دوال تقوم بتنفيذ عمليات على الدوال المخزنه

- **ان اى برنامج داخل #C قد يحتوى على اكثر من namespace على الاقل لابد من وجود واحد فقط وقد يحتوى على اكثر من Class لكن على الاقل يجب ان يحتوى على Class واحد**
- **وبعد System هو namespace الرئيسى بصفته يحتوى على classes تحتوى على دوال التوصيف للدخل والخرج داخل اى برنامج**

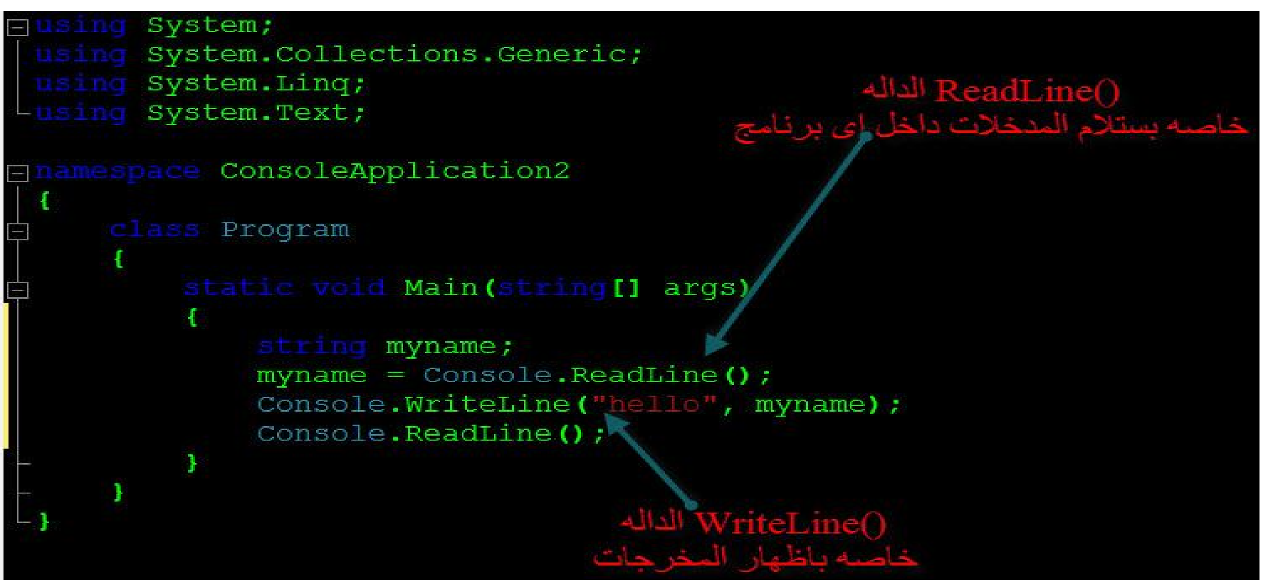
- ان وظيفة الداله الرئيسيه ()Main ان Compiler يبدء دائما بتنفيذ خطوات البرنامج من عندها ايا كان موضعها ولابد ان تكون موجوده داخل class الرئيسى للبرنامج والذى يعطى اسم الافتراضى Program ومن الممكن وضع تلك الداله داخل اى class وسعنتبره انه الرئيسى الذى يحتوى على تلك الداله

- ان لغة #C حساسه لحالة الاحرف بمعنى انه يجب ان يراعى فيها capital او small

- **شرح الاوامر البرمجيه داخل لغة #C : -**

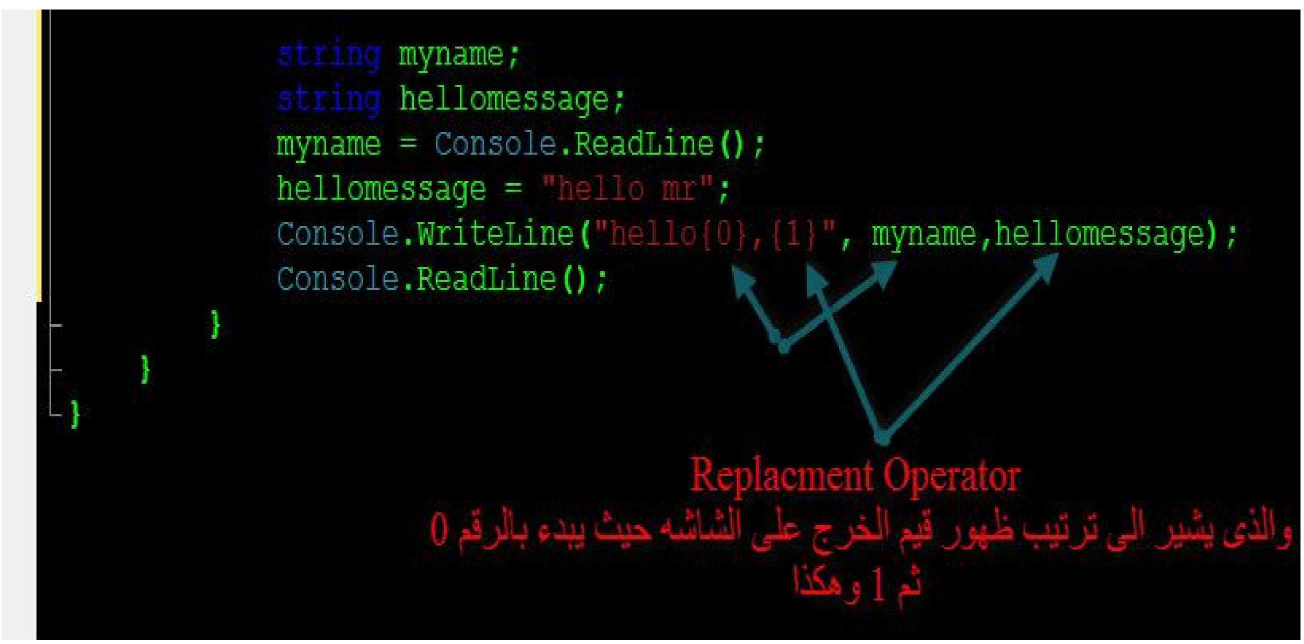
- **اوامر الطباعه :**

هناك امران اساسيان داخل #C وهما كيفية التعامل مع الدخل والخرج والذى لاغنى عنهم داخل اى برنامج وهما ممثلان فى دالتين يتبعان نفس class



دوال الدخل والخرج والتى تتبع جميعها التصنيف Console	
Write()	**هذه الداله تقوم بطباعة الخرج ويظل المؤشر فى مكانه**
WriteLine()	**هذه الداله تقوم بطباعة الخرج وينتقل المؤشر الى سطر جديد اسفل الخرج**
ReadLine()	**تقوم هذه الداله بقراءة المدخلات ودائما ما تعتبر لغة السى شارب ان جميع المدخلات على انها نصوص لذلك يجب تحويل المدخلات بعد قرأتها**

- معامل تنظيم ظهور الخرج على الشاشه وما يعرف <u>Replacement Operator</u>

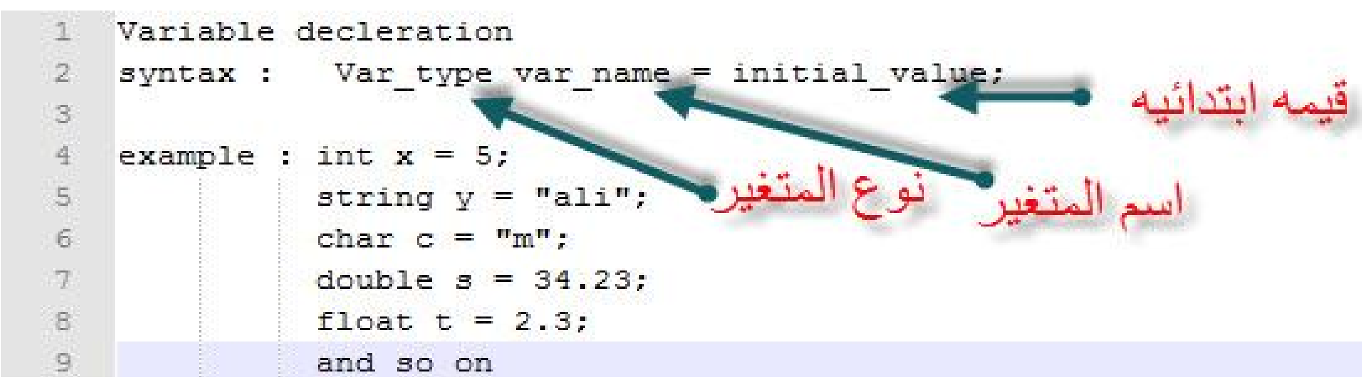


انواع اليانات داخل لغة C# :

انواع البيانات داخل السى شارب وسعة كلا منها

النوع	فيما يستخدم	سعته
Byte	للارقام	0 الى 8 bits
Short	للارقام	2byte الى bits16
Int	للارقام	32byte to 4byte
Char	للحروف	8byte for one char
Float	للكسور	4byte
Boolean	للقيم المنطقيه	True or false
String	للنصوص	Stream open of char
Double	للارقام التى تحتوى علامات عشريه	8byte

القاعده العامه لتعريف المتغيرات داخل C# :



داخل لغة السى شارب لابد من الاعلان عن المتغيرات صراحة وتعريفها قبل استخدامها

او اعطاؤها اى قيمه

```
using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {

            string myname;
            string hellomessage;
            myname = Console.ReadLine();
            hellomessage = "hello mr";
            Console.WriteLine("hello{0},{1}", myname,hellomessage);
            Console.ReadLine();
        }
    }
}
```

تم تعريف المتغيرات قبل استخدامها او اعطاؤها قيم

يأخذ المتغير القيمه من المستخدم اثناء تنفيذ البرنامج

اعطاؤه قيمه ابتدائيه

دوال التحويل داخل لغة السى شارب :

قلنا سابقا ان السى شارب تعتبر جميع المدخلات على انها نصوص فكيف اذا ما تم ادخال ارقام او انواع اخرى فهناك دوال للتحويل بين انواع البيانات

دوال التحويل داخل السى شارب	
Convert	Parse

ولدينا الامثله التاليه :



تستطيع الان استخدام نفس الدالتين لتحويل باقى انواع البيانات حيث انك مضطر

لاستخدامها فى جميع الاحوال

- انواع المعاملات داخل السى شارب :

Arithmetic operators	**المعاملات الرياضيه**
Multiplication	*
Addition	+
Subtraction	-
Module	%
Dividing	/
Increment by 1	++
Decrement by 1	--

Competitive Operators	**معاملات المقارنه**
Smaller than	<

Greater than	>
Equal	==
Smaller than or equal	<=
Greater than or equal	=>
Not equal	!=

Logical operator	**المعاملات المنطقيه**
And	**&&**
Or	||
not	**!**

معاملات المساواه Assignment	
Assignment plus	+=
Assignment minus	-=

ان جميع المعاملات التى تم سردها فى الجداول السابقه لها الاستخدام الاوسع

داخل السى شارب وتدخل فى معظم العمليات الحسابيه و المنطقيه

Flow Control Statments

استخدام الدوال الشرطيه داخل اللغه Flow Control – statements :

الدوال الشرطيه من الدوال الواسعة الانتشار داخل لغات البرمجه بصفه عامه وذلك لاهميتها الكبيره فى التحكم فى سير البرنامج ولدينا العديد منها داخل السى شارب .

القاعده الشرطيه IF

القاعده العامه لها :

```
IF Statment : syntax : if (condition)
                             {
                             statments ;
                             }
```

بمعنى لو تحقق الشرط سوف يتم تنفيذ الجمل البرمجيه التى تتبع القاعده الشرطيه

```
        static void Main(string[] args)
        {
            int mynumber;
            mynumber = int.Parse(Console.ReadLine());
            if (mynumber == 10)
            {
                Console.WriteLine("we are win the race");
            }
        }
    }
}
```

الشرط المطلوب تحقيقه

الجمله التى ستنفذ اذا تحقق الشرط

القاعده الشرطيه IF ELSE :

القاعده العامه لها :

```
IF Elase Statment : syntax : if (condition)
                             {
                             statments ;
                             }
                             Else
                             {
                             statments;
                             }
```

بمعنى انه اذا تحقق الشرط سيتم تنفيذ الجمل التابعه IF واذا لم يتم تحقيق الشرط سيتم تنفيذ الجمل التابعه ELSE

```
        static void Main(string[] args)
        {
            int mynumber;
            mynumber = int.Parse(Console.ReadLine());
            if (mynumber == 10)
            {
                Console.WriteLine("we are win the race");
            }
            else
            {
                Console.WriteLine("we are lose");
            }

        }
    }
}
```

الجمله الاخرى فى حالة عدم تحقيق الشرط

القاعده الشرطيه NESTEAD IF :

القاعده الشرطيه المتداخله والصوره العامه لها :

```
Nestaed IF Statment : syntax

If (condition1)
      If (condition2)
         {
               Statements1;
            }
Else if (condition3)
         If (condition4)
            {
                  Statments2;
Else
            Statments3;
Else
         Statments4;
```

لاحظ دائما فى حالة if المتداخله فان else تتبع اقرب if لها

```
    {
        static void Main(string[] args)
        {
            int mynumber;
            mynumber = int.Parse(Console.ReadLine());
            if (mynumber <= 10)
                if (mynumber >= 5)        فى حالة تحقيق الشرطين تطبع الجمله الاولى
                {
                    Console.WriteLine("we win the race");
                }
                else                      فى حالة عدم تطبيق الشرط الثانى بعد الاول تطبع
                {                         الجمله الثانيه
                    Console.WriteLine("we are lose");
                }
            else                          فى حالة عدم تطبيق اى من الشرطين تطبع الجمله الثالثه
            {
                Console.WriteLine("wrong value");
            }
            Console.ReadLine();
```

القاعده الشرطيه SWITCH CASE :

القاعده العامه لها :

```
Switch Case statment : syntax
Switch (variable or expertion)
{
Case 1: {statements};
Break;
.
.
Case n :{ statements};
Break;

Default :{ statements};
Break;
}
```

يتم تعريف المتغير ثم فى كل مره اعطاؤه قيمه معين مع كل جملة Case وتنفيذ الجمله المطلوبه اذا ما تساوت هذه القيمه مع اى حاله من تلك الحالات ولا حظ انه بعد كل حاله لابد من وضع الكلمه المحجوزه Break لتفصل كل حاله عن الاخرى ثم فى النهايه وضع الحاله Default والتى سوف تنفذ جملتها اذا لم تتحقق اى من الحالات السابقه

```
string thename;
thename = Console.ReadLine();
switch (thename)
{
    case "ali":
        Console.WriteLine("this is my name");
        break;
    case "mohamed":
        Console.WriteLine("this is my brother name");
        break;
    case "basem":
        Console.WriteLine ("this is my frend name");
        break ;
    default :
        Console.WriteLine("i dont know that name");
        break;
 }
Console.ReadLine();
```

كل حاله منفصله عن الاخرى

ستنفذ هذه الجمله اذا لم يتوفر اى شرط من الشروط السابقه

Loops statments

الجمل التكراريه LOOPS :

تستخدم هذه الجمل فى تكرار جزء معين من البرنامج اذا ما تحققت شروط معينه حيث يكون التكرار محدود وله ضوابط وليس الى مالانهايه ولها انواع داخل لغات البرمجه وسنوضح الان كل جمله وطريقة عمله

الجمله التكراريه For Loop :

القاعده العامه لها :

```
For Statment :syntax
For (initialize; condition; update);
{Statements};
```

```
using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 10; i++) ;

            int k;
            for (k = 0; k < 10; k++)


            Console.ReadLine();
```

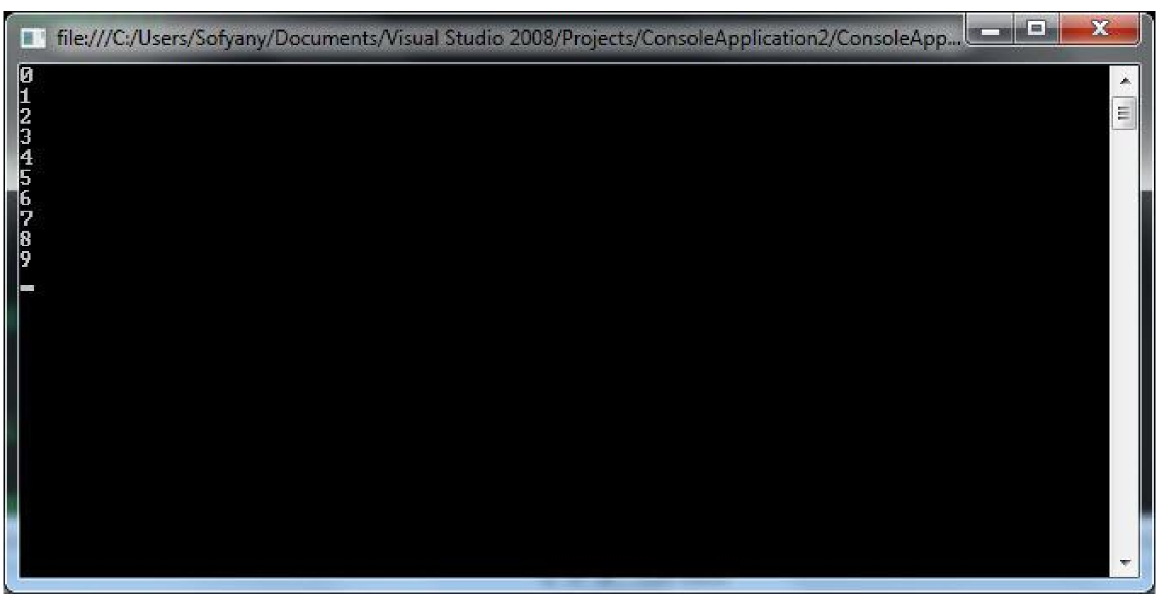
اولا يوضع القيمه الابتدائيه ثم الشرط ثم
التحديث بحيث لا يزيد التحديث عن الشرط
المحدد ولاحظ انه يمكن تعريف المتغير
داخل جملة التكرار

الحاله الاخرى تعريف المتغير مسبقا ثم بناء الجمله عليه

```
        static void Main(string[] args)
        {
            int i;
            for (i = 0; i < 10; i++)
            {
                Console.WriteLine(i);
            }
            Console.ReadLine();

        }
    }
}
```

الجمله التى ستنفذ فى كل مره يتحقق
فيها الشرط وهى طباعة قيمة المتغير



القاعده الشرطيه While Loop :

وهذه القاعده تؤكد على استمرار تنفيذ جزء معين من الكود طالما ان الشرط مازال متحققا والقاعده العامه لها كالتالى

```
While Loop statment :syntax

        Initialize;
While (condition)
{
Statements;
Update;
}
```

```
static void Main(string[] args)
{
    int i = 0;
    while (i < 10)
    {
        Console.WriteLine(i);
        i++;
    }

    Console.ReadLine();

}
    }
}
```

القيمه الابتدائيه

الشرط

الجمله

التحديث وبعد كل مره يعود ليختبر الشرط مع while

```
0
1
2
3
4
5
6
7
8
9
```

الناتج

القاعده الشرطيه DO While :

و هذه القاعده لها خصوصيه حيث انه يتم تنفيذ الكود اول مره قبل اختبار الشرط بمعنى انه اول مره سيتم التنفيذ حتى لو يتحقق الشرط وبعد ذلك يختبر الشرط كل مره . القاعده العامه لها :

```
do while statment : syntax
Initialize;
do
{
Statements;
Update;
}
While (condition);
```

لاحظ ان الشرط وضع فى نهاية الجمله حتى يتم التنفيذ فى المره الاولى اذا لم يتحقق لكن بعد ذلك فأنه لن يتم النفيذ الا اذا تحقق الشرط

```
static void Main(string[] args)
{
    int i = 0;
    do
    {
        Console.WriteLine(i);
        i++;
    }
    while (i < 10);

    Console.ReadLine();

}
}
}
```

المثال الاول سوف يطبع القيمه الاولى ثم يختبر الشرط بعدها


0
1
2
3
4
5
6
7
8
9
الخرج


static void Main(string[] args)
{
int i = 11;
لاحظ هنا ان القيمه الابتدائيه تخالف الشرط
do
{
Console.WriteLine(i);
i++;
}
while (i < 10);
Console.ReadLine();
}
}
}


11
ومع ذلك قام بطباعة
القيمه الاولى وتوقف عند
ذلك لانه اختبر الشرط
فى المره الثانيه ووجده

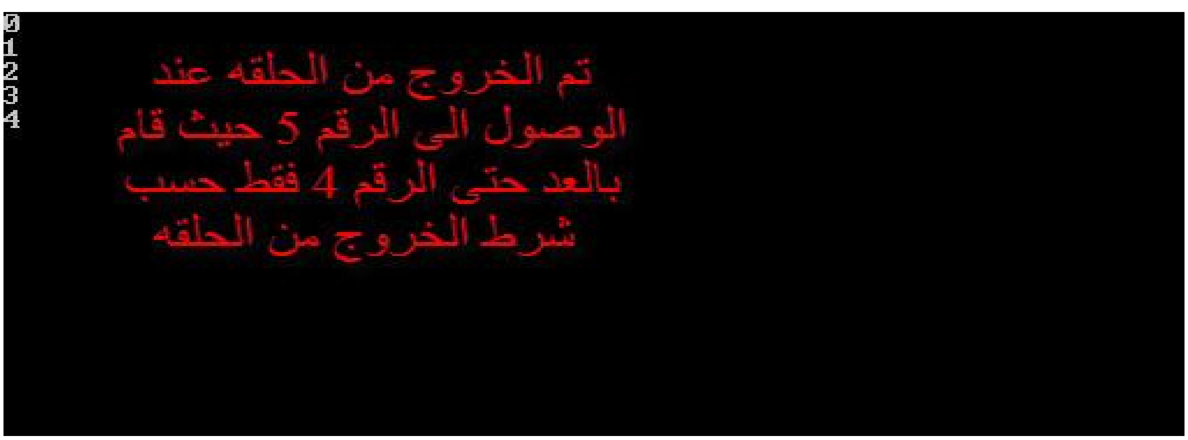
جملة الخروج عن الحلقة التكراريه عند الوصول الى نقطه معينه break :

والتى تقوم بانهاء الحلقه عند الوصول الى نقطه معينه او تحقيق شرط معين

القاعده العامه لها : تستخدم داخل اى من الجمل التكراريه السابقه ودائما توضع

بعد اختبار الشرط كل مره ولدينا المثال التالى :



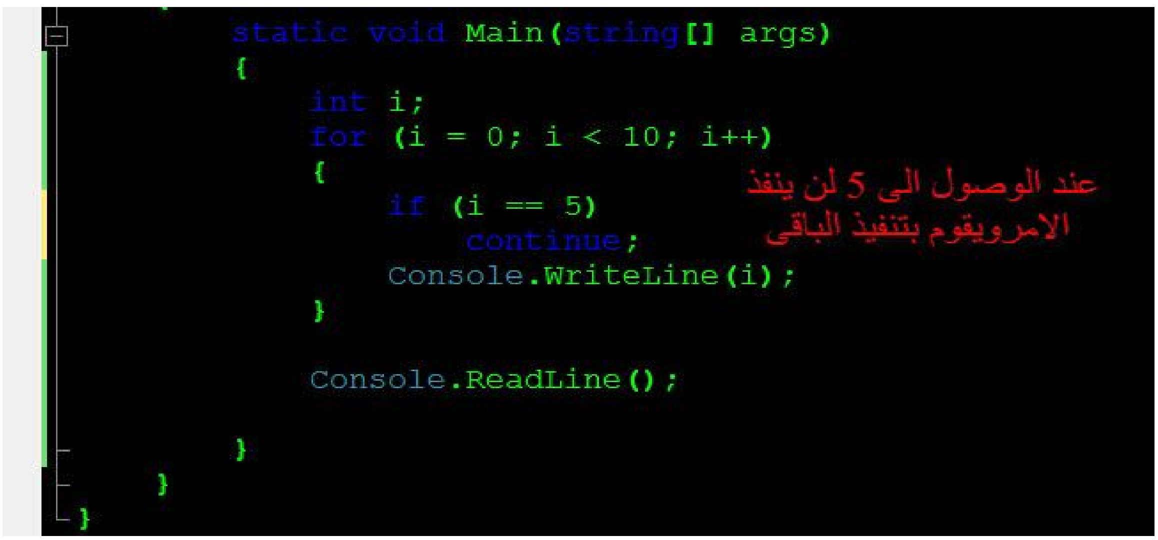


امر الاستمرار داخل الحلقه عند تحقيق شرط معين Continue :

هذا الامر يخرج من الحلقه ثم يعود اليها مره اخرى عند شرط معين بمعنى انه يتخطى تنفيذ جزء معين من الحلقه ثم يعود لينفذ الباقى

ولدينا المثال السابق حيث سيتجاهل البرنامج طباعة الرقم 5 ثم يطبع باقى الارقام

القاعده العامه لها : انها توضع فى نفس مكان break





Compound Data: Arrays

1 – المصفوفات او Array :

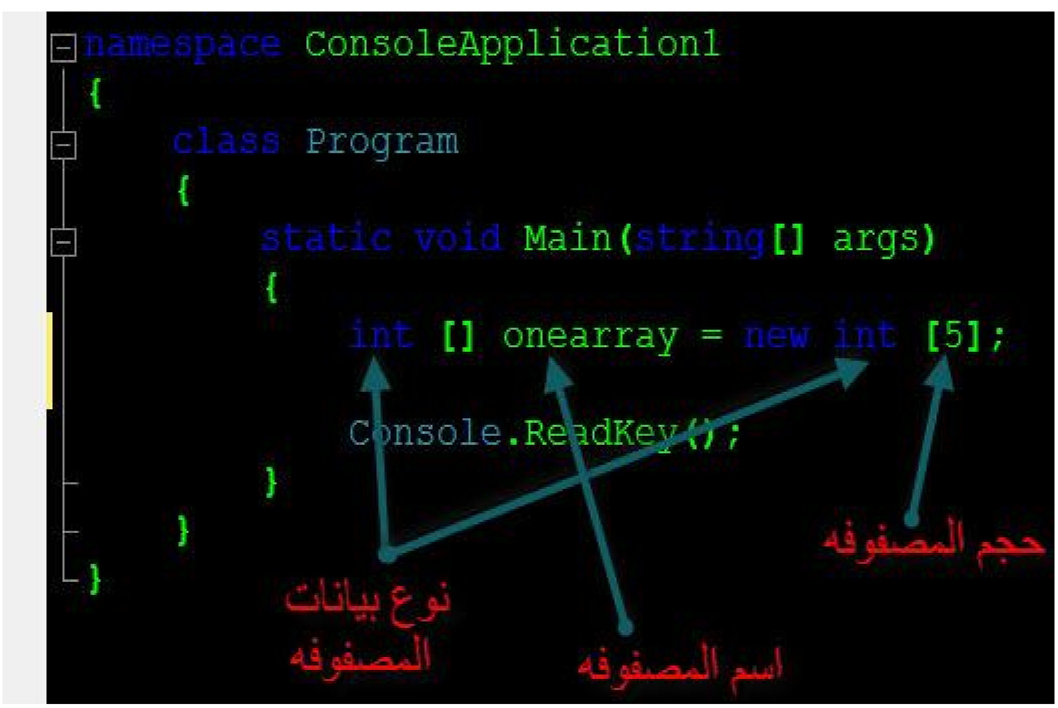
يجب ان يكون كل عناصر المصفوفه لها نفس نوع البيانات وممكن ان تختلف فى القيم

ويوجد لها نوعان المصفوفه احادية البعد او one –D والمصفوفه الثنائية البعد او

Tow – D

القاعده العامه لتعريف المصفوفه الاحدايه :

```
Array Definition : syntax
Data type [] reference name = new data type [size];
```



ومن الممكن ان تعطيها قيما ابتدائيه عند التعريف لكن يجب الاتزيد القيم عن حجم المصفوفه

```
class Program
{
    static void Main(string[] args)
    {
        int [] onearray = new int [5]{10,15,20,25,30};

        Console.ReadKey();
    }
}
```

القيم الابتدائيه من نفس نوع البيانات ولاتزيد عن حجم المصفوفه

ومن الممكن ايضا تحديد حجم المصفوفه عن طريق القيم الابتدائيه لكن ليست مستحبه

ومن الممكن اعطاء قيما ابتدائيه للمصفوفه من المستخدم وقت تنفيذ البرنامج

- طريقة الاتصال مع المصفوفات الاحادية :

```
Write data to element in array :
Direct in code: array name [index of element] = value;
X [2] = 10;

write data from user on runtime:
 X [2] = int.parse (Console.ReadLine ());

print element data :
Console.WriteLine(x [2]);

modifi on element data:
X [3] = x [3]*3;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] x = new int[10];
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine("enter no :{0}", i);
                x[i] = int.Parse(Console.ReadLine());

            }

            Console.ReadKey();
        }
    }
}
```

سعة المصفوفه 10 لذلك لن يزيد الحلقه التكراريه عن ذلك

سوف يتم استلام بيانات المصفوفه من المستخدم وقت تنفيذ البرنامج

ومن المثال السابق سوف بتكملة البرنامج لحساب اكبر قيمه واقل قيمه تم ادخالها من المستخدم كما يلى :

```
    {
        static void Main(string[] args)
        {
            int i;
            int[] x = new int[10];
            for ( i = 0; i < 10; i++)
            {
                Console.WriteLine("enter no :{0}", i);
                x[i] = int.Parse(Console.ReadLine());
            }
            int max = x[0];
            int min = x[0];
            for (int j = 0; j < 10; j++)
            {
                if (x[j] < min)
                    min = x[j];
                if (x[j] > max)
                    max = x[j];
            }
            Console.WriteLine("the max no :{0}", max);
            Console.WriteLine("the min no :{0}", min);
            Console.ReadKey();
        }
    }
```

هنا تم تعريف متغيرين بالقيمه الصغرى والعظمى
واعطائهم القيمه المواجوده بالعنصر الاول من
المصفوفه

تتم المقارنه العنصر الاول من المصفوفه بجميع العناصر الاخرى
ونرى من فيهم الاصغر من الاخر حتى يتم ايجاد الاصغر فيهم
جميعا ثم بعدها نفس العمليه تتكرر من فيهم الاكبر من الاخر حتى
نجد الاكبر فيهم ثم نقوم بطباعة القيمتين على الشاشه

: For Each …..Loop

هناك نوع اخر مهم جدا من الحلقات التكراريه وغالبا ما يستخدم مع المصفوفات حيث يستخدم فى قراءة بيانات المصفوفه ولا يمكن استخدامه فى الكتابه اليها او تغير قيمها

القاعده العامه لها :

```
Foreach statment : syntax
For each (<data type> <identifier> in array>)
{Stmts};
```

```
static void Main(string[] args)
{
    int i;
    int[] x = new int[10];
    for ( i = 0; i < 10; i++)
    {
        Console.WriteLine("enter no :{0}", i);
        x[i] = int.Parse(Console.ReadLine());
    }
    foreach (int m in x)
        Console.WriteLine(m);

    Console.ReadKey();
}
}
}
```

استخدام الحلقه التكراريه foreach فى قراءة محتويات المصفوفه وطباعتها

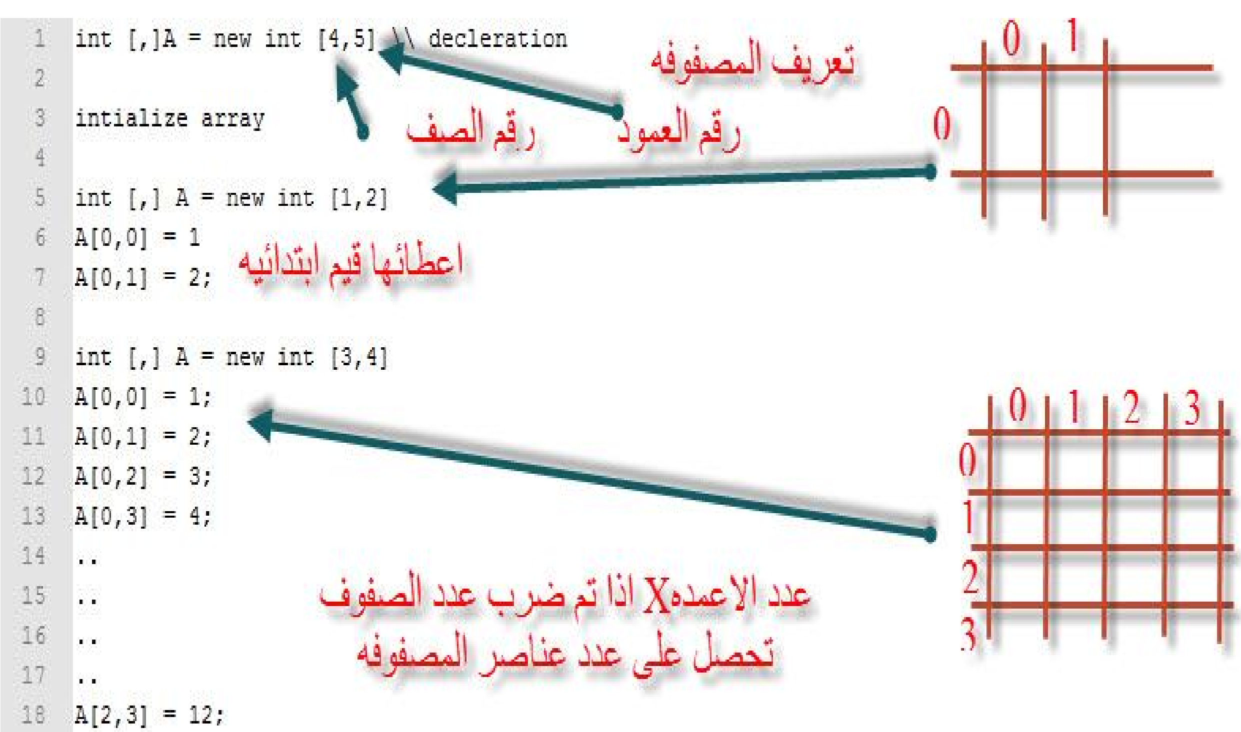
: Tow – D Array

المصفوفات ثنائية الابعاد هى عباره عن نوع من المصفوفات يشبه الجداول حيث يتكون من صفوف واعمده ويكون الخلايا وهذه الخلايا هى عناصر المصفوفه

التوصيف العام لها :

```
2-D array : syntax
Data type [,] arrayreferance = new data type [rows_no, columns_no];
```

For example:



```
int [,]A = new int [4,5] \\ decleration

intialize array

int [,] A = new int [1,2]
A[0,0] = 1
A[0,1] = 2;

int [,] A = new int [3,4]
A[0,0] = 1;
A[0,1] = 2;
A[0,2] = 3;
A[0,3] = 4;
..
..
..
..
A[2,3] = 12;
```

اعطاء بيانات للمصفوفه من المستخدم اثناء تنفيذ البرنامج فى هذا المثال :

```
static void Main(string[] args)
{
    int[,] x = new int[4, 5];
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            Console.WriteLine("enter value of element:({0},{1})", i, j);
            x[i, j] = int.Parse(Console.ReadLine());
        }
    }

    Console.ReadKey();
}
}
}
```

حلقة تكراريه للصفوف

حلقة تكراريه للاعمده

كتابة عنوان العنصر

ادخال البيانات

```
enter value of element:(0,0)
1
enter value of element:(0,1)
2
enter value of element:(0,2)
3
enter value of element:(0,3)
4
enter value of element:(0,4)
5
enter value of element:(1,0)
6
enter value of element:(1,1)
7
enter value of element:(1,2)
8
enter value of element:(1,3)
9
enter value of element:(1,4)
10
enter value of element:(2,0)
11
enter value of element:(2,1)
12
enter value of element:(2,2)
13
enter value of element:(2,3)
14
enter value of element:(2,4)
15
enter value of element:(3,0)
16
enter value of element:(3,1)
17
enter value of element:(3,2)
18
enter value of element:(3,3)
19
```

تم اعطاء قيم للعناصر حتى 20 عنصر

ولقراءة عناصر المصفوفه :

```
Read 2d array values

Console.WriteLine(A[2,3]);
```

```
20
(0,0) value is : 1
(0,1) value is : 2
(0,2) value is : 3
(0,3) value is : 4
(0,4) value is : 5
(1,0) value is : 6
(1,1) value is : 7
(1,2) value is : 8
(1,3) value is : 9
(1,4) value is : 10
(2,0) value is : 11
(2,1) value is : 12
(2,2) value is : 13
(2,3) value is : 14
(2,4) value is : 15
(3,0) value is : 16
(3,1) value is : 17
(3,2) value is : 18
(3,3) value is : 19
(3,4) value is : 20
```

طباعة عناصر المصفوفه
الثنائيه

وللتعديل على بيانات المصفوفه الثنائيه :

```
Modifi 2-d array

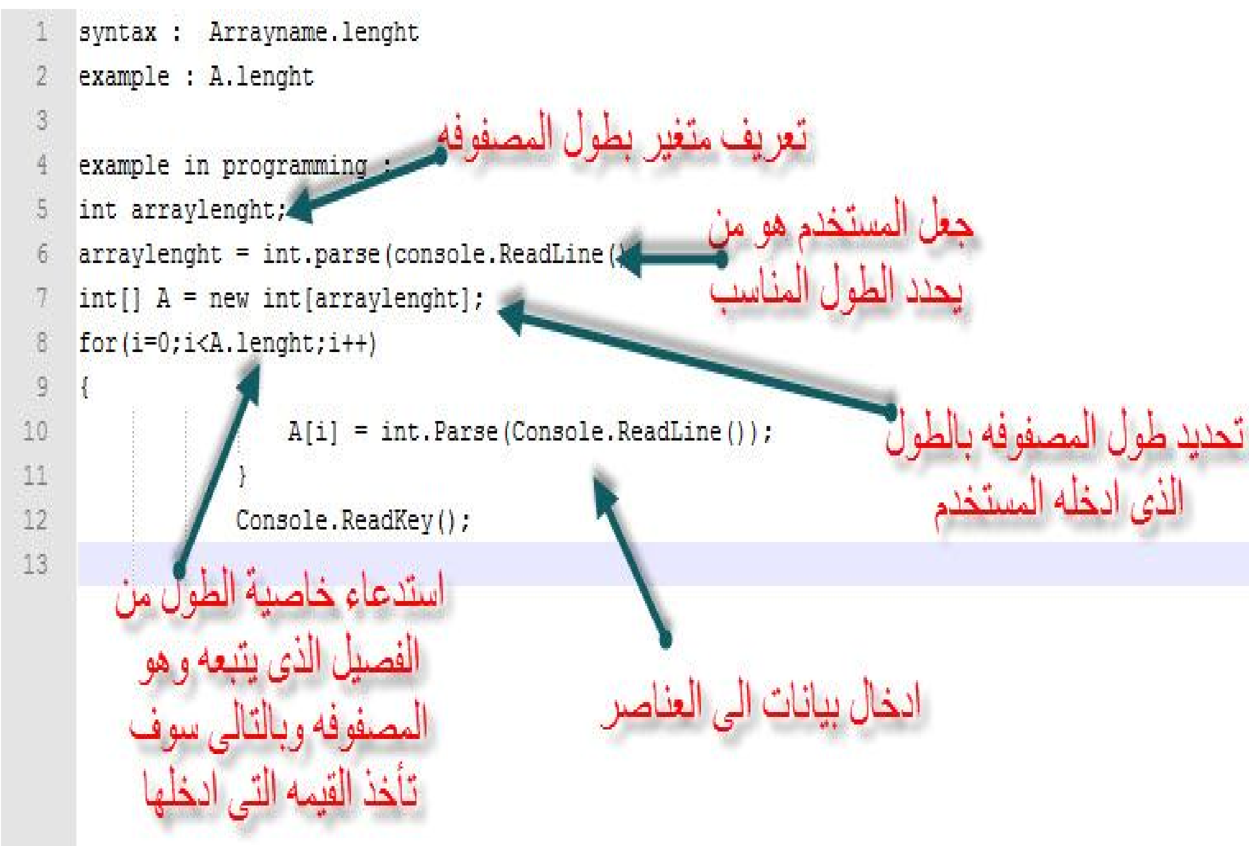
A[0,0] = A[0,0] * 2
```

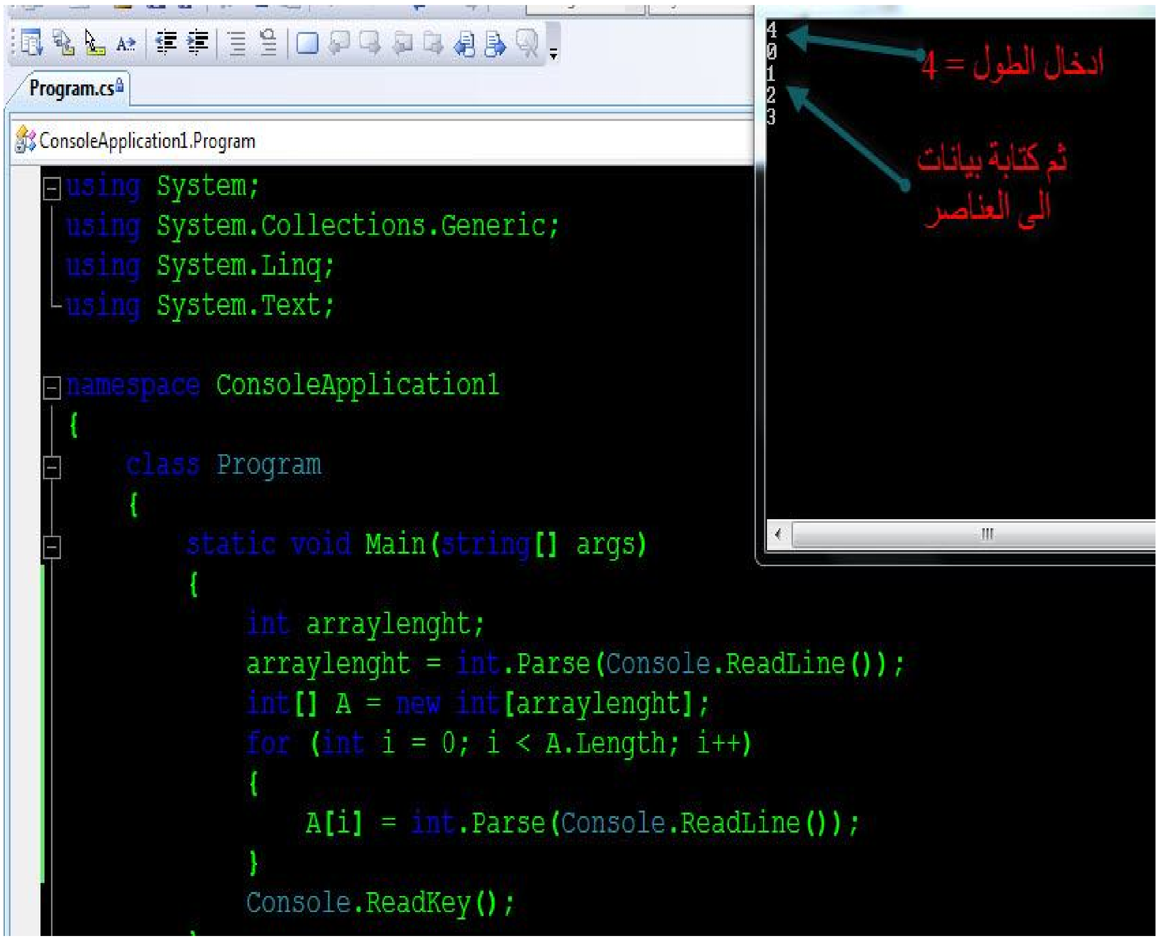
: Array Class Properties And Methods

الخصائص والدوال التى يتضمنها الفصيل Array

Array Length – 1

ومنه تعرف طول المصفوفه ةتفيدك فى حالة اذا كان عدد العناصر مجهولا لك بمعنى انه عند تعريف المصفوفه فلن يتم تحديد طول معين بل سيقوم المستخدم بأدخاله وقت التنفيذ





2 – الخاصيه Rank :

والتى منها تعرف نوع المصفوفه اذا كانت ثنائية الابعاد ام احادية البعد

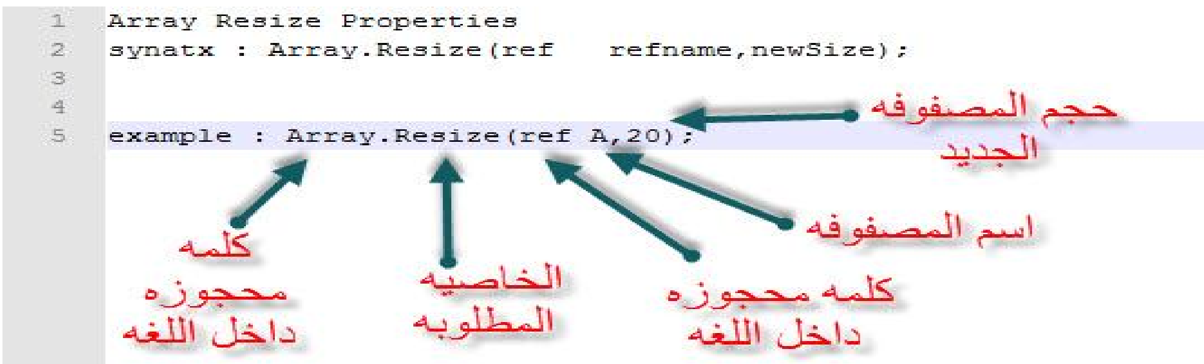
واستدعائها يكون كالتالى :

```
Rank Properties

syntax : arrayname.Rank
```

3 – الخاصيه التى تسمح بتغير حجم المصفوفه Resize :

تغيير الحجم مع الاحتفاظ بالقيم القديمه لها والقاعده العامه لاستدعائها



4 – خاصية البحث داخل عناصر المصفوفه Binary Search :

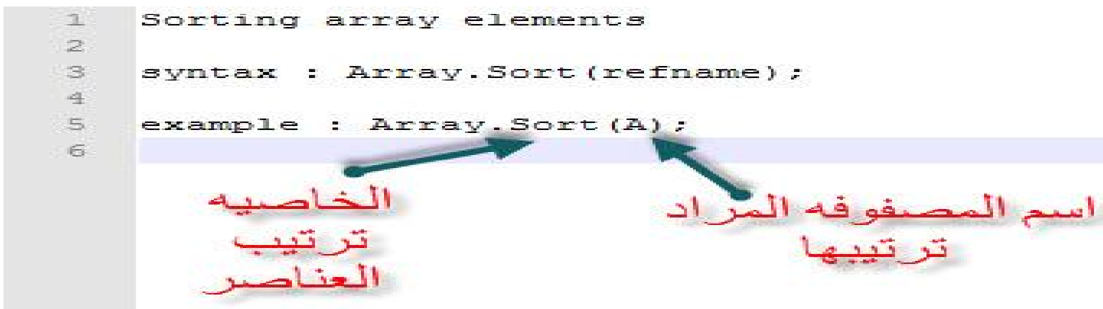
وتقوم هذه الخاصيه بالحبث داخل عناصر المصفوفه عن عنصر يحتوى على قيمه معينه لاستخدامه ولكن شرط ان تكون عناصر هذه المصفوفه مرتيه ترتيبا تنازليا او تصاعديا

القاعده العامه لاستخدام الخاصيه :



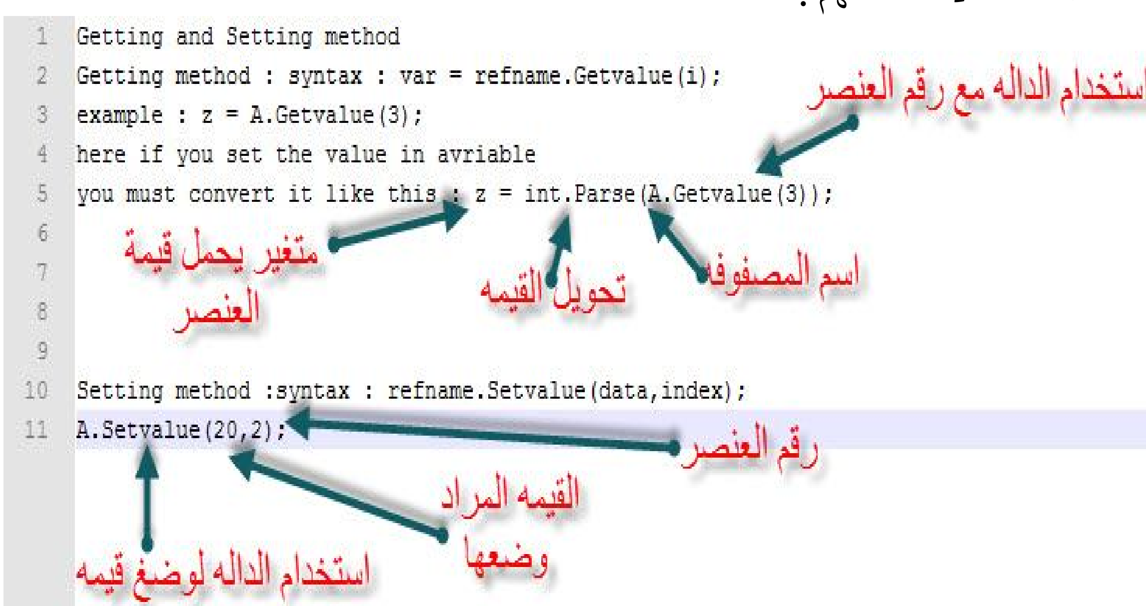
5 – الخاصيه التى تسمح بترتيب عناصر المصفوفه Sort :

القاعده العامه لاستخدامها :



6 – الداله Getting لاحضار قيمة العناصر والداله Setting لوضع قيم لعناصر المصفوفه :

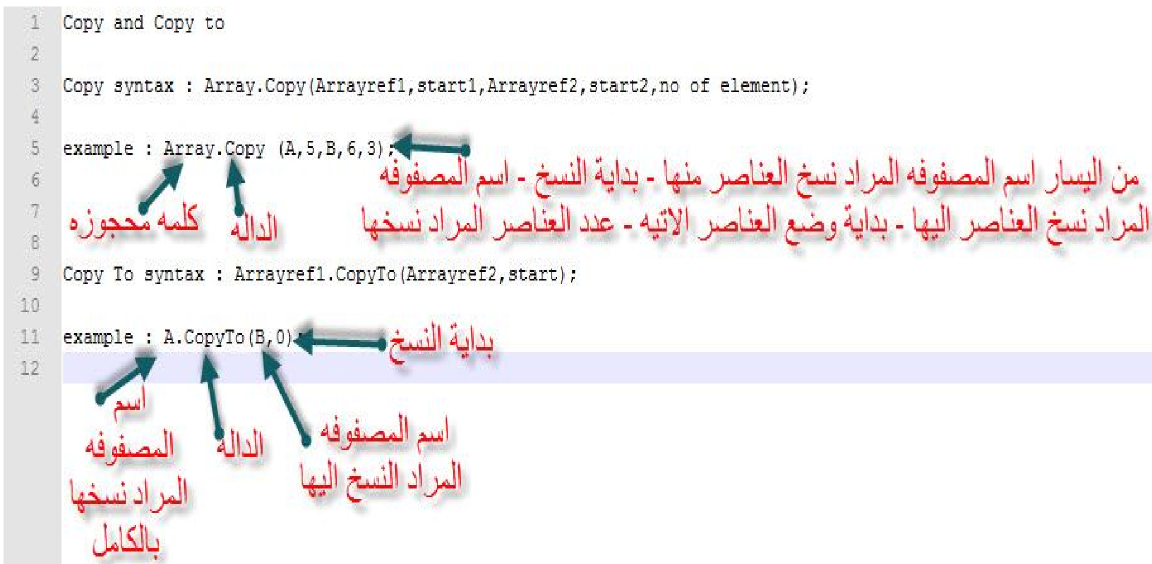
القاعده العامه لاستخدامهم :



7 – دوال نسخ المصفوفات الى اخرى وهما نوعان

1 – Copy وتقوم بنسخ جزء من مصفوفه الى اخرى
2 – Copy To تقوم بنسخ كامل مصفوفه الى اخرى

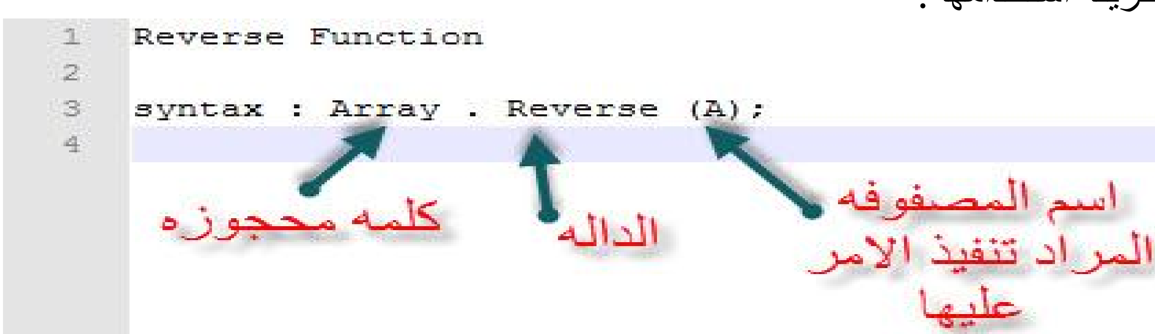
قاعدة استخدامهم :



8 – الداله Reverse :

والتى تقوم بعكس عناصر المصفوفه اى عكس وضعية العناصر

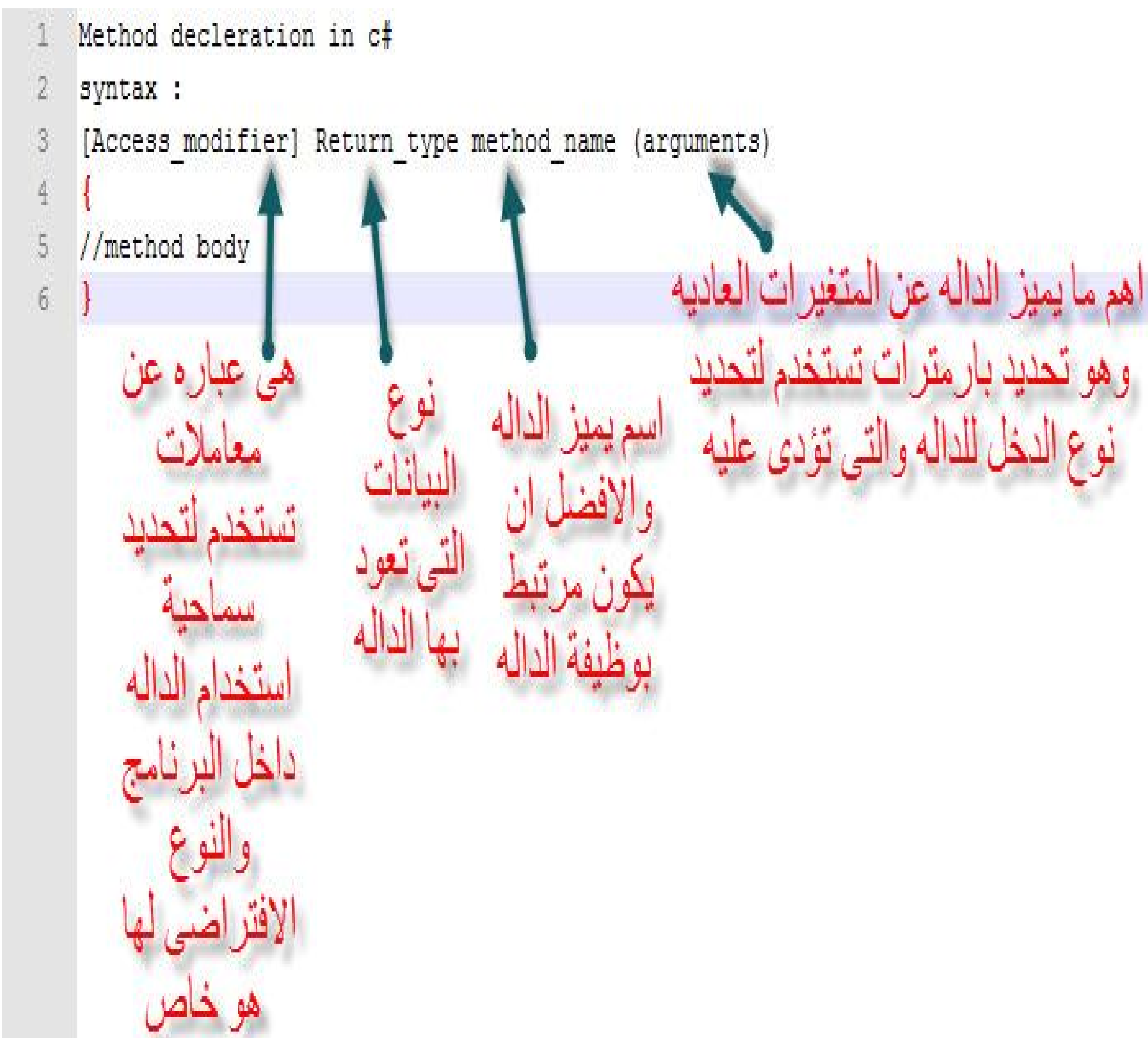
طريقة استخدامها :



C# Methods

الدوال واستخدامها داخل السى شارب :

القاعده العامه لاستخدام الدوال داخل لغة #C :



انواع المعاملات او Access Modifiers :

```
Private :
Public :
Protected :
```

لها فقط class يسمح هذا النوع بالتعامل مع الداله او المتغير داخل

class يسمح هذا النوع بالتعامل مع الداله او المتغير خارج

يسمح هذا النوع بالتعامل مع لاداله او المتغير داخل المشروع فقط namespace لديه او

- بعض الامثله على بناء الدوال وتعريفها داخل البرنامج :

```
examples :

int fact (int number);
{method code}


int bar ();
{method code}


void sum (int x, int y)
{method code}
```

داله لكن لها بارامتر واحد فقط

داله ليس لها اى بارامترات

داله لها بارمتران لكن ليس لها قيمه عائده منها بسبب المعرف void حيث انه لا يسمح باى قيمه عائده من الداله

هذه هى الثلاثة احتمالات التى تبنى عليها الداله داخل السى شارب وهناك احتمال رابع وهو نادر وهو ان الداله لاتعود بقيمه وليس لها بارامترات ومن امثلة ذلك الداله التى تستخدم فى طباعة الخرج على الشاشه WriteLine()

بعض الامثله :


{
class Program
{
void showdate()
{
DateTime thedate;
thedate = DateTime.Now;
Console.WriteLine(thedate);
}
static void Main(string[] args)
{
Program pro = new Program();
pro.showdate();
Console.ReadKey();
}
}
}
داله لاظهار تاريخ اليوم ليس لها
بارامترات او قيمه عائده
3/28/2013 10:22:49 PM
الخرج
ولاستدعنها يجب تخليق كائن جديد من
الفصيل التابعه له ثم استدعائها بهذا الكائن


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
class Program
{
static int getarea(int height, int width)
{
return height * width;
}
static void Main(string[] args)
{
Console.WriteLine(getarea(4, 4));
Console.ReadKey();
}
}
}
داله لها بارامتران وتعود بقيمة المساحه
file:///C:/Users/Sof
16
الخرج
استدعاء الداله واعطائها قيمة
البارامترات

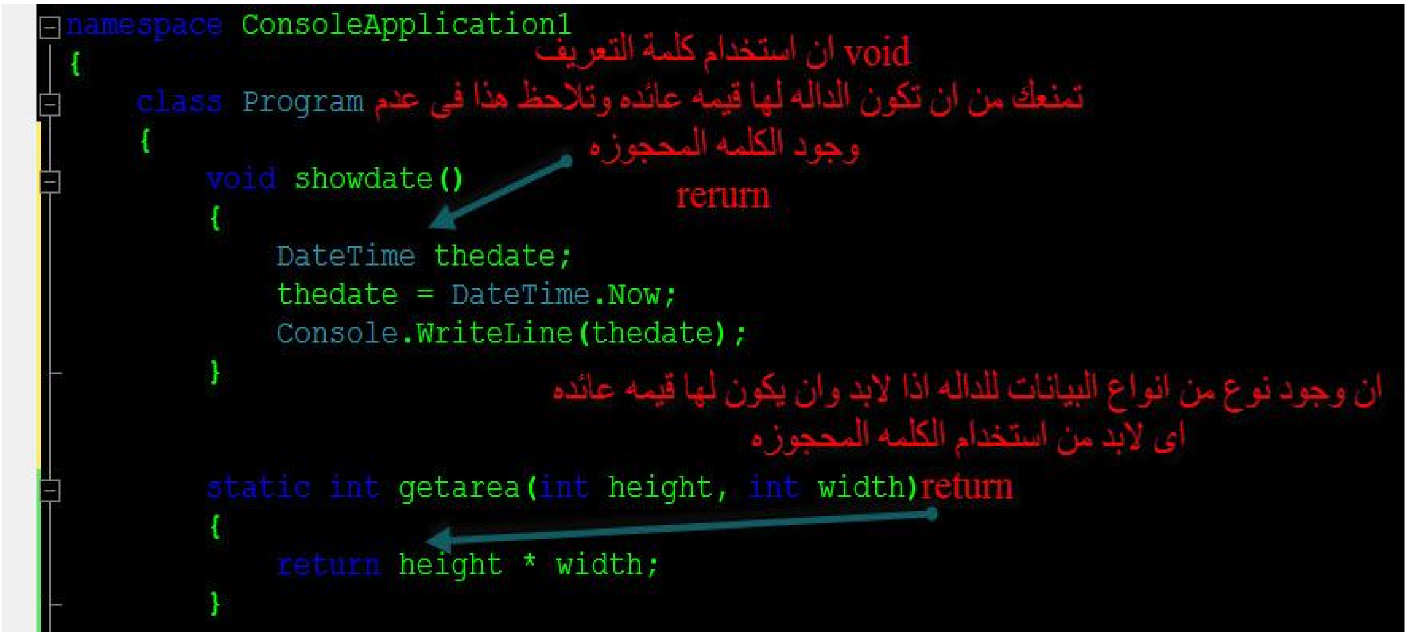
ومن المثالين السابقين نريد ان نسلط الضوء على بعض الاساسيات الهامه جدا فى استخدام الدوال :

1 – ان استدعاء الدوال يكون على حسب المعرف لها او Access Modifier وقد قمنا بشرح انواعه .

2 – ان كتابة الداله داخل الفصيل او Class قبل الداله الرئيسيه Main او بعدها ويكون استدعائها من داخل الداله Main او من اى داله اخرى .

3 – ان استدعاء اى داله لا يكون الا عن طريق انشاء كائن جديد من الفصيل او Class والذى تتبعه الداله او الذى تم انشائها بداخل كواحده من عناصره الاساسيه ولا يكون غير ذلك الا فى حاله واحده وهى ان تعطى المعرف Static وهو الوحيد الذى يسمح بأستدعاء الداله بشكل منفرد دون الحاجه الى كائن المخلق من الفصيل ويفسر هذا وضع هذا المعرف دائما قبل الداله الرئيسيه Main داخل اى برنامج فى السى شارب حتى يتسنى استدعائها من داخل البرنامج مباشرتا لان التنفيذ يبداء من عندها

4 – ان الداله او قد يكون لها اكثر من بارامتر او لا يكون لديها اى بارامترات وقد تعود بقيمه واحده او اكثر من قيمه او لاتعود بقيم على الاطلاق لكن يجب ان تلاحظ الفرق بين الحالتين فى المثال التالى :



ان الكلمتان void و return لايجتمعان فى تركيب داله واحده

5 – من الممكن اعطاء القيم التى تعود بها الدوال لمتغيرات لكن من نفس نوع البيانات التى تعود به الداله

6 – لابد من اعطاء كافة القيم للبارامترات التى تحملها الداله داخل الكود ولابد ان تكون القيم من نفس نوع تلك البارامترات

- فى المثال التالى نقوم بعمل برنامج يقوم ببعض العمليات الرياضيه لكن باستخدام الدوال :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{ class Program
    { static double sum(int x, int y)
        {
            return x + y;
        }
        static double sub(int x, int y)
        {
            return x - y;
        }
        static double mul(int x, int y)
        {
            return x * y;
        }
        static void Main(string[] args)
        {
            int x, y;
            Console.WriteLine("enter value of x and y");
            x = int.Parse(Console.ReadLine());
            y = int.Parse(Console.ReadLine());
            Console.WriteLine(sum(x, y));
            Console.WriteLine(sub(x, y));
            Console.WriteLine(mul(x, y));
            Console.ReadKey();}
    }
}
```



طرق اعطاء البارامترات الى الدوال Passing Parameters to method :

1 – by value

وهى ارسال القيمه الى البارامتر مباشرتا وتكون عباره عن نسخ Data الاساسيه

ونقلها الى مكان فى الذاكره وبالتالى هذا استغلا سيىء للذاكره لانه ياخذ مساحه

اكبر بدون داعى ومثال عليه :

```
using System.Linq;
using System.Text;

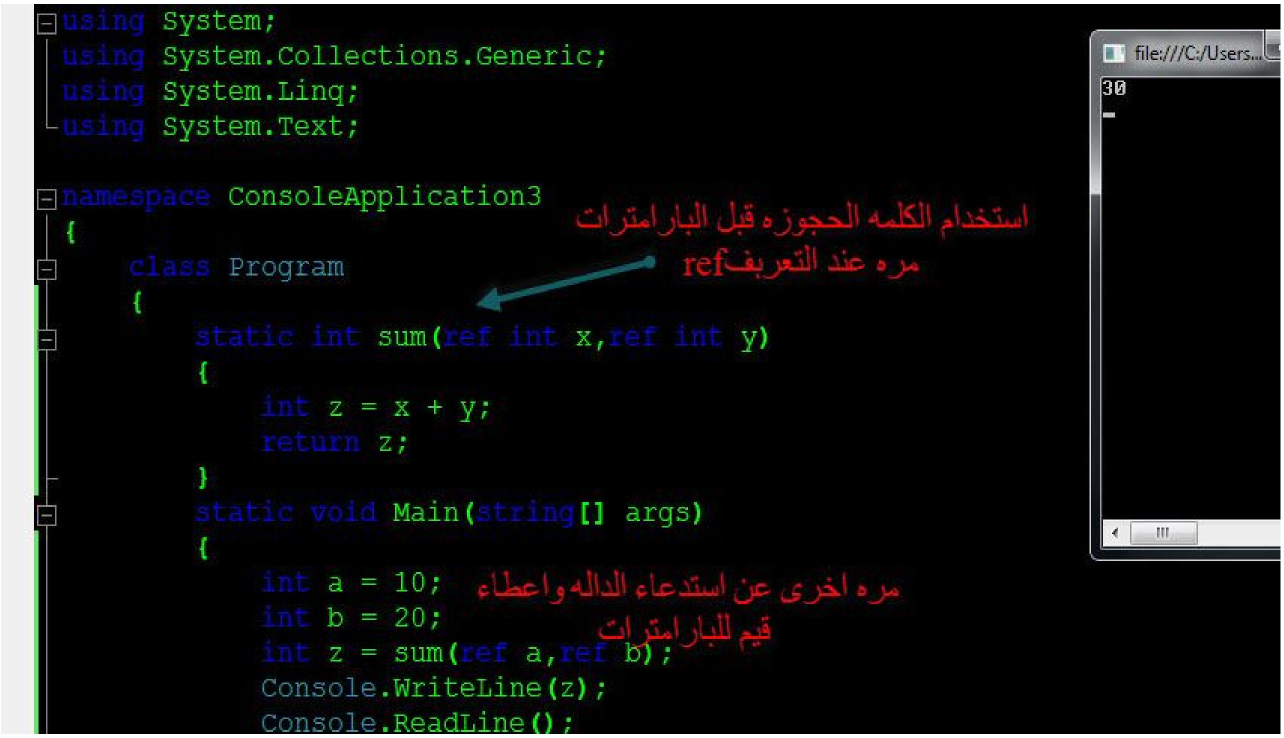
namespace ConsoleApplication3
{
    class Program
    {
        static int sum(int x, int y)
        {
            int z = x + y;
            return z;
        }
        static void Main(string[] args)
        {
            int a = 10;
            int b = 20;
            Console.WriteLine(sum(a, b));
            Console.ReadLine();
```

```
30
```

2 – by reference

الطريقه الثانيه وهى اعطاء البارامترات عن طريق الاشاره الى مكان معين فى الذاكره

واليك المثال التالى :



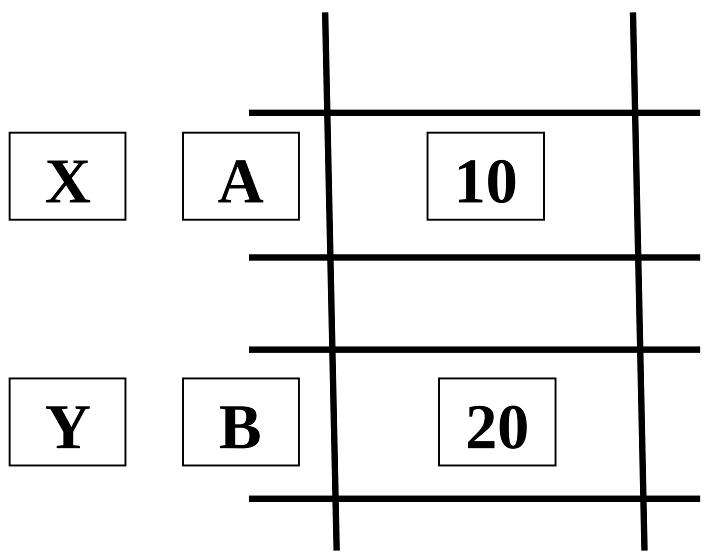

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static int sum(ref int x,ref int y)
        {
            int z = x + y;
            return z;
        }
        static void Main(string[] args)
        {
            int a = 10;
            int b = 20;
            int z = sum(ref a,ref b);
            Console.WriteLine(z);
            Console.ReadLine();
```

ان استخدام هذه الطريقه هو الاستخدام الامثل للذاكره لانها كلا من البارامترات

والمتغيرات التى تمثلها تشير الى مكان واحد فى الذاكره

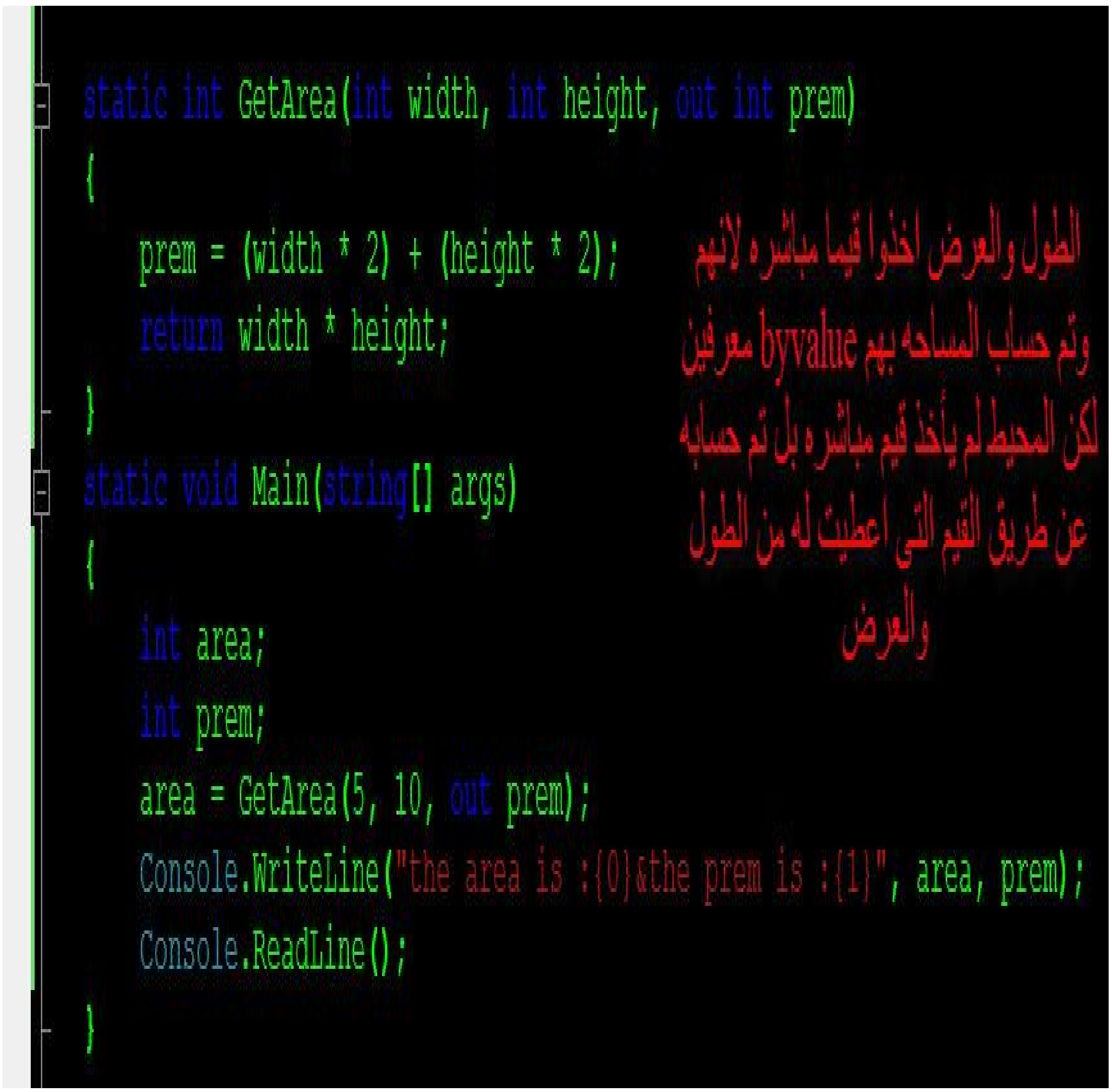
لاحظ ان استخدام
REF
لابد من اعطاء قيم للبارامترات قبل استدعاء الداله اى انك تدخل البيانات ثم تقوم باستدعاء الداله لتجمعهم



by out – 3

هى نفس استخدام ref لكن الاختلاف فى انك تستدعى الداله اولا ثم تعطيها قيما

ابتدائيه ولنى فى المثال التالى :



: C# Variable Scopes

النطاق الذى تستخدم فيه المتغيرات داخل اللغه :

وله ثلاثة انواع :

class level – 1

متغيرات يتم تعريفها داخل class ويتم استخدامها مع اى method اخرى

method level – 2

متغيرات خاصه ب method ولاتستطيع استخدامها مع اى method اخرى

nested level – 3

متغيرات تعرف داخل اى block من data ولا تستخدم خارج هذا block مثل الحلقات مثلا او الجمل الشرطيه

: Passing Arrays as Parameters

Passing by value – 1

برنامج عمل array ويضع قيمه ابتدائيه ويعطيها الى method لكى تطبعها

على الشاشه وهنا سوف نعطى المصفوفه للداله كبارامتر

```
using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static void printarray(string[] w)
        {
            for (int i = 0; i < w.Length; i++)
            {
                Console.Write(w[i]);
                Console.WriteLine();
            }
        }
        static void Main(string[] args)
        {
            string[] weekdays = new string[]
            { "sat", "sun", "mon", "tus", "wen", "thru", "fri" };
            printarray(weekdays);
            Console.ReadLine();
```

```
sat
sun
mon
tus
wen
thru
fri
```

: Method overloading

من الممكن بناء اكثر من داله بنفس الاسم لعمل عدة وظائف تحت مسمى واحد بشرط ان تختلف هذه الدوال فى عدد البارامترات او ان البارامترات تختلف نوع البيانات المعرفه بها وتسمى هذه العمليه Overloading او التحميل الذائد

```
{
    class Program
    {
        public int add(int a, int b)
        {
            return a + b;
        }
        public float add(float a, float b)
        {
            return a + b;
        }
        public int add(int a, int b, int c)
        {
            return a + b + c;
        }
        static void Main(string[] args)
        {
            Console.WriteLine(add(2, 3));
            Console.WriteLine(add(2.2, 3.3));
            Console.WriteLine(add(2, 3,4));
            Console.ReadLine();

        }
    }
```

جميع الدوال بنفس الاسم لكنها
تختلف من ناحية تعريف
البارامترات ونوع بياناتها

Object Oriented Programming

- رؤيه فلسفيه حول البرمجه الكائنية التوجه :

ان مفهوم البرمجه الكائنية التوجه لهو بالاساس يعتمد على النظره المجرده للحياه التى نعيشها ذلك بأن كل شيء فى هذه الحياه عباره عن مجموعه من الكائنات ولها صفاتها الخاصه ومميزاتها والشكل التى تتصرف به وتجد ان الطبيعه قد ربطت بين كل الاشكال المختلفه للحياه كل منهم له اسلوبه الخاص بالتعامل واسلوبه الخاص فى الطريقه التى يتعامل بها مع الاخر وذلك الربط ادى فى النهايه الى تكوين فضاء يحتوى هذه الكائنات جميعا لكى توجه الى فكرة العوالم المتوازيه مبرمجه طبيعيا لان تتعامل مع بعضها فى تعاون برمجى كامل وبالتالى ارادو تطبيق تلك الفكره والسير على خطاها فى مفهوم البرمجه الكائنية التوجه فهو يعطى فكره مجرده للنظام ككل ثم يقسمه الى مجموعه من الكائنات التى لها خصائص تميزها عن غيرها ويوضع مع كل كائن الاساليب التى يتعامل بها مع غيره .

- مزايا البرمجه كائنية التوجه :

1 – سهولة متابعة وصيانة السوفت وير

2 – سرعة تطوير البرمجيات ومتانتها

Object Oriented Programming Concepts

1 – Encapsulation

ان مجموعه من العناصر لها نفس الخصائص ولها نفس السمات من الممكن ان تجتمع داخل فصيل واحد ويسمى Class وهذا الفصيل هو الذى يمثلها بوجه عام لكن لا يتعامل معها عن طريقه بل عن طريق كائن مخلق منه وهو Object

- اذا ما اردنا ايجاد امثله من الحياه لدينا Class يمثل البشريه وهذا الفصيل له جميع صفات البشريه واسلوب تعاملها مع غيرها والذى هو فصيل رئيسى من منظومه اشمل وسوف يطلق عليها namespace وهذه المنظومه ممثله فى الكون اذا اردنا انشاء كائن من فصيل البشريه ليكن الانسان الذى يحيا على الارض واى انسان اذا يحمل جميع صفات وسلوكيات هذا الفصيل

ومن الممكن ان ننشيء فصيل اخر من فضاء الكون وليكن فصيل للحيوان

ومن الممكن ان يكون هناك فصيل للنباتات واخر للجماد وهكذا

: Class Builder Features

مالذى توفره لك Classes من مزايا داخل البرمجه كائنية التوجه ؟

1 – استطعت بذلك ان تقوم بتمثيل كل Object او عنصر داخل System بمجموعه من Classes والتى تحتوى على كل التاعاريف لهذا Object من سمات وتصرفات

2 – استطعت حماية بياناتك الاساسيه لهذا Object من محاولة العبث بها او حتى تغييرها والتى تتمثل فى المفاهيم الاتيه :

1 – Encapsulation Features مزايا التغليف :

```
Encapsulation Builder :
يبنى الفصيل بواسطة الكلمه المحجوزه التى تعرفه ثم يعطى اسم ثابت له
class   class_name
{
//attrbuites//   تكتب هنا المتغيرات او
int var1;        ما يسمى بالمعلومات
int var2;        التى تخص هذا الفصيل
string var3;

public string function_name(parameters)
{
//method body//   توضع هنا الدوال التى تعبر عن تصرفات الفصيل والتى تكون ثابته خلال حياة النظام
}

public data_type property_name
{
get
{                          توضع هنا الخواص التى يعتمد عليها
return attrbuites_value;   الفصيل والتى تعطى قيما متغيره وهذه
}                          الخواص هى التى تعطيك التحكم فى
set                        قيم لمعلومات الفصيل عند بعد
{
attrbuites = value;
}
}
}
```

مزايا التغليف :

1 – State Retention

الابقاء على حاله وتصرفات العناصر او Object داخل System وذلك يمثل فى الجزء الذى يعطى له وهو Methods فقط انا اعطى لها القيم وهى تقوم بالتصرف الثابت لها

2 – Data Hiding

يتم عزل البيانات والتى تكون ممثله فى Attributes للعنصر Object داخل النظام بعيدا عن اى تغيير

- ويبقى هنا السؤال كيف لى ان استخدام هذا الفصيل الضخم والمليىء بعدة عناصر وكيف لى ان استخدام تلك العناصر من خارج هذا الفصيل ؟

وتاتى الاجابه بأن بناء الفصيل شيء واستخدام محتوياته شيء اخر بمعنى ان الفصيل الذى يسمى البشريه هو فصيل عام يطلق على بنى البشر لكن يجب ان يجسد فى عنصر معين لكى تستطيع ان تتعامل معه وهذا العنصر الذى يمثله هو الانسان وبالتالى سوف يتم انشاء عنصر من كل فصيل لكى تستطيع ان تستخد كافة عناصره

انتهت النظره الفلسفيه لناتى الى تديعمها بأمثله عمليه :

- سماحية استخدام عناصر Class داخل البرنامج :

تحتوى لغة #C على خمسة انواع منهم :

private : وتعطى سماحية استخدام عناصر الفصيل داخل الفصيل فقط

protected: داخل الفصيل او فصيل مشتق منه

internal: يستخدم فى اى مكان داخل المشروع عامة

protected internal : استخدامه داخل المشروع او مشروع مشتق منه

public: يتعامل مع عناصر الفصيل من اى مكان داخل نفس المشروع او داخل فضاء الاسماء المضمن بداخله

بعض الامثله على بناء Classes داخل C# :

```
using System.Text;

namespace ConsoleApplication3
{
    class EX1
    {
        private int x;
        private int y;

        public void setdata(int a, int b)
        {
            x = a;
            y = b;
        }
        public int sum()
        {
            return x + y;
        }
        public float avg()
        {
            return sum() / 2;
        }
    }
    class Program
```

تعريف الفصيل بأسم معين

تعريف المتغيرات من نوع خاص اى لا يمكن استخدامها الا داخل الفصيل فقط

تعريف داله من نوع عام والتى يمكن استدعائها خارج الفصيل والتى تعطى قيما للمتغيرات

تعريف داله للجمع

تعريف داله للمتوسط الحسابى

والان كيف لنا ان نستخدم عناصر هذا الفصيل داخل البرمجه ؟

والجواب هو ان نقوم بانشاء كائن يتبع هذا الفصيل وبالتالى فهو يستطيع استدعاء كافة عناصره كالتالى : القاعده البرمجيه

```
create object from class

class_name  obj_name = new class_name;
```

اسم الفصيل مره اخرى

كلمه محجوزه تدل على انشاء الكائن

اسم الفصيل الذى تريد انشاء كائن منه

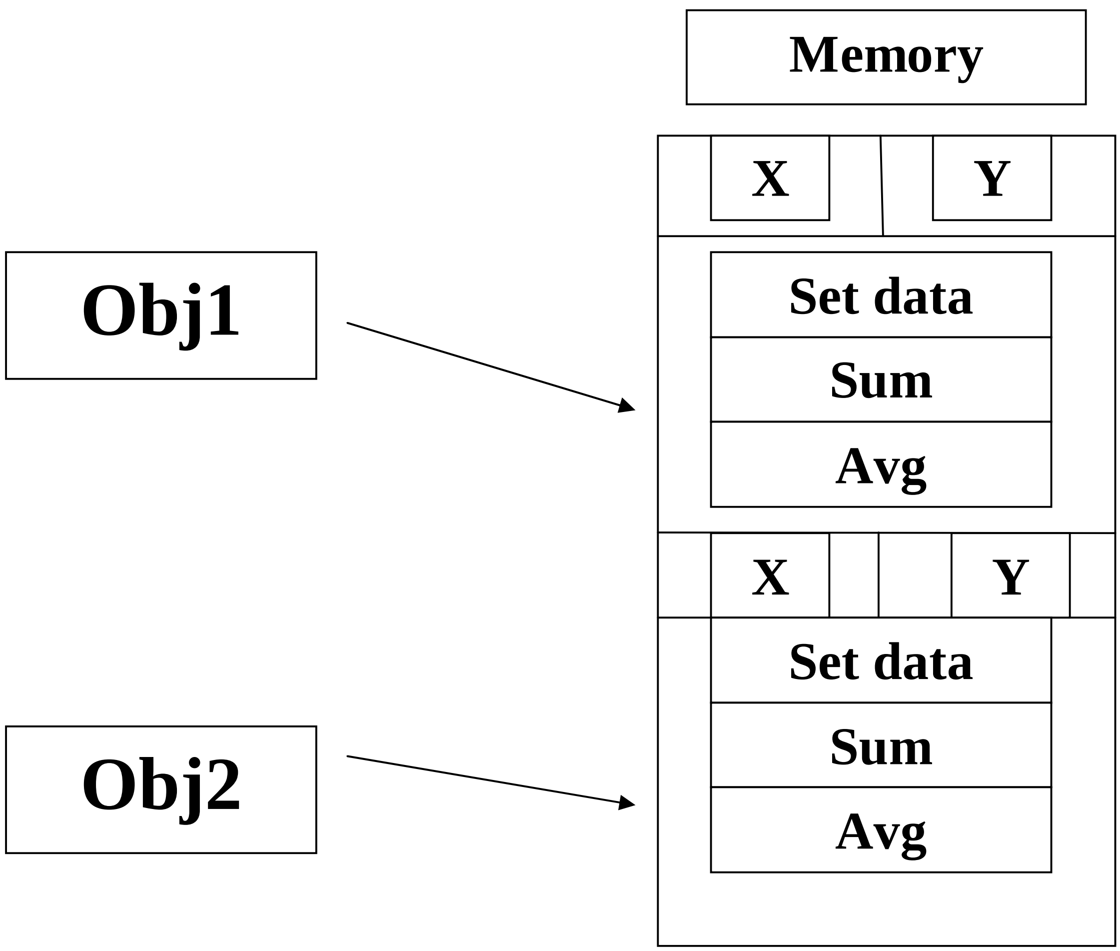
اسم الكائن الذى تريد انشاؤه

```
class Program
{

    static void Main(string[] args)
    {
        EX1 obj1 = new EX1();
        EX1 obj2 = new EX1();
        Console.ReadLine();

    }
}
}
```

انشاء الكائن وتعريفه يكون داخل الفصيل الرئيسى داخل
الداله الرئيسيه ومن الممكن تعريف اكثر من كائن من
نفس الفصيل ولهذا ميزه كبيره وهى ان عناصر الكائنين
لا يتاثروا ببعضهم البعض رغم انهم ينتمون لفصيل واحد
لكن لكل منهم مكانه فى الذاكره



توصيات هامه : يفضل ان تكون المتغيرات داخل الفصيل تعطى المعرف Private حتى لا يتم العبث بها او تغيرها .

يفضل ان تعطى الدوال داخل الفصيل المعرف Public حتى تستطيع استخدامها خارج الفصيل .

- كيفية الاتصال باعضاء class باستخدام الكائن الجديد:

القاعده البرمجيه :

```
Accessing object members:

object_name . member_name;
```

اسم الكائن   اسم العضو

سنقوم فى المثال السابق باعطاء قيم لعناصر الكائنين الذى تم تعريفهم من نفس الفصيل ونرى النتيجه لدالتين الجمع والمتوسط الحسابى :

```
static void Main(string[] args)
{
    EX1 obj1 = new EX1();
    EX1 obj2 = new EX1();

    obj1.setdata(5, 10);
    obj2.setdata(15, 20);
    Console.WriteLine(obj1.sum());
    Console.WriteLine(obj1.avg());
    Console.WriteLine(obj2.sum());
    Console.WriteLine(obj2.avg());
    Console.ReadLine();
```



- هناك دالتين رئيستين تعطى قيما ثابته Private members وهذا يحدث مع اى لغه تستخدم البرمجه كائنية التوجه حيث تستخدم دالتين :

1 – Setter : لاعطاء القيم الابتدائيه لاعضاء class

2 – Getter : لقراءة البيانات واظهارها على الشاشه

بعض الامثله على ذلك :

```
private string name;

//getter function :
public string getname()
{
return name;
}
//setter function
public void setname(string thename)
{
name = thename;
}
```

هذه الدوال من الممكن استخدامها داخل السى شارب لكن اللغه لها اسلوب متطور وهو استخدام الخواص او properties والتى تقوم بنفس المهمه بشكل اكثر كفائه ويتوافق مع النظريه الكائنية التوجه

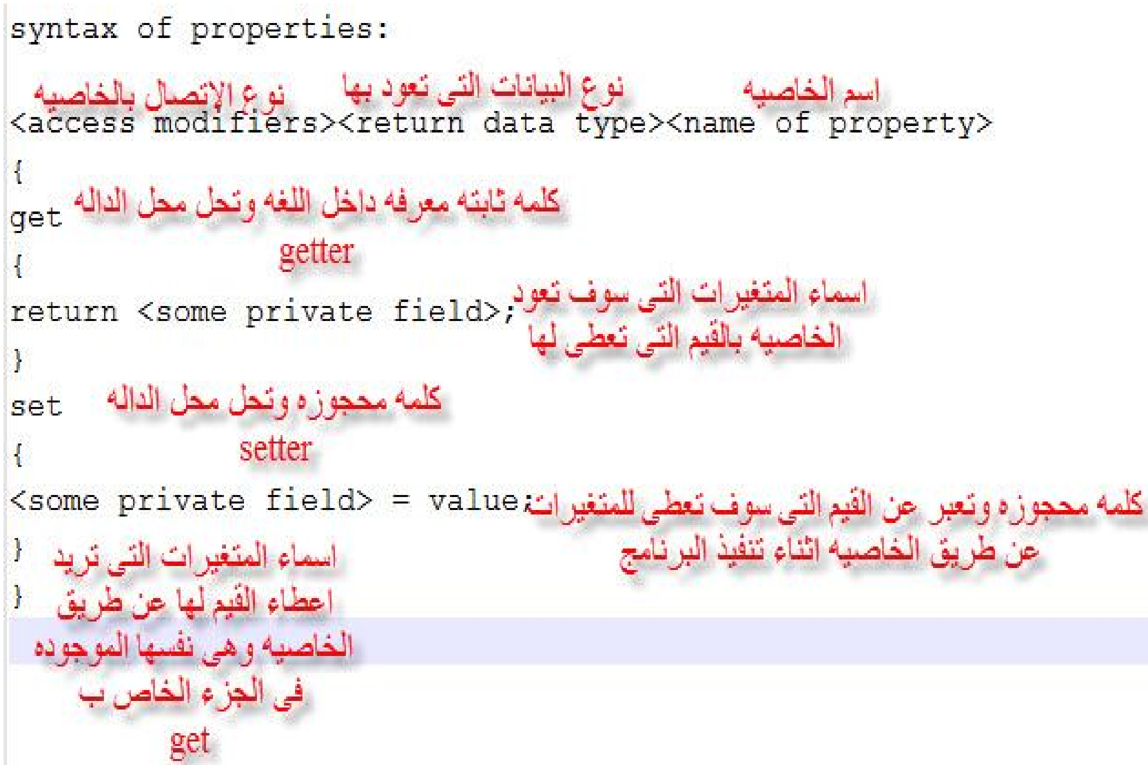
- انواع الاداه Properties :

1 – Read Only

2 – Write only

3 – Read / Write

القاعده العامه لتعريفها داخل Classes :



بعض التوصيات :

الخواص التى تبنى داخل classes ليس لها اى بارامترات

اسم الخاصيه لابد ان يكون هو نفسه اسم المتغير التى تتعامل معه مع الفارق بان يكون

اول حرف من اسم الخاصيه capital حتى يتميز عن المتغير

ولنرى بعض الامثله على بناء الخواص :

```
class example
{
    int num1, num2;
    public int Num1
    {get{ return num1; }
        set{ num1 = value; } }
    public int Num2
    {
        get { return num2; }
        set { num2 = value; }
    }
    public int sum()
    { return num1 + num2; }
    public float avg()
    { return sum() / 2; }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        example obj1 = new example();
        obj1.Num1 = 20;
        obj1.Num2 = 30;
        Console.WriteLine(obj1.Num1);
        Console.WriteLine(obj1.Num2);
        Console.WriteLine(obj1.sum());
        Console.WriteLine(obj1.avg());
        Console.ReadLine();

    }
}
}
```

اعطاء القيم عن طريق الخواص

اظهار القيم عن طريق الخواص

اظهار نواتج الدوال



لاحظ ان القيم اعطيت للمتغيرات عن طريق الكلمه set داخل الخاصيه وتم اظهار البيانات عن طريق الكلمه get

: Static Member

ان هذا المعرف له ميزه خاصه جدا داخل السى شارب حيث انه يقوم بمشاركة العضو الذى يعرف بواسطته بين جميع الكائنات التى تعرف من نفس الفصيل ويلزم لتعريف



ولنرى مثال عام :

```
{
    class student
    {
        public static int phonenumber;
        public int idnumber;
        public string name;
    }
    class Program
    {
        static void Main(string[] args)
        {
            student st1 = new student();
            student st2 = new student();
            st1.idnumber = 3;
            st1.name = "ahmed";
            st2.idnumber = 4;
            st2.name = "kamel";
            student.phonenumber = 492067;
        }
    }
}
```

تعريف العضو المشارك عليه من جميع الكائنات

استدعاء جميع الاعضاء بالقاعده الاساسيه وهى باستخدام الكائنات

عدا رقم التليفون فقد تم استدعاؤه عن طريق الفصيل مباشرتا

ان اى static member ينتمى الى الفصيل مباشرتا وليس الكائن وهذه ميزه

هامه جدا داخل السى شارب

دوال البناء والهدم داخل السى شارب :

1 – دالة البناء Constructor :

هى داله خاصه باى Class وتستخدم لوضع القيم الابتدائيه لعناصره

خصائصها :

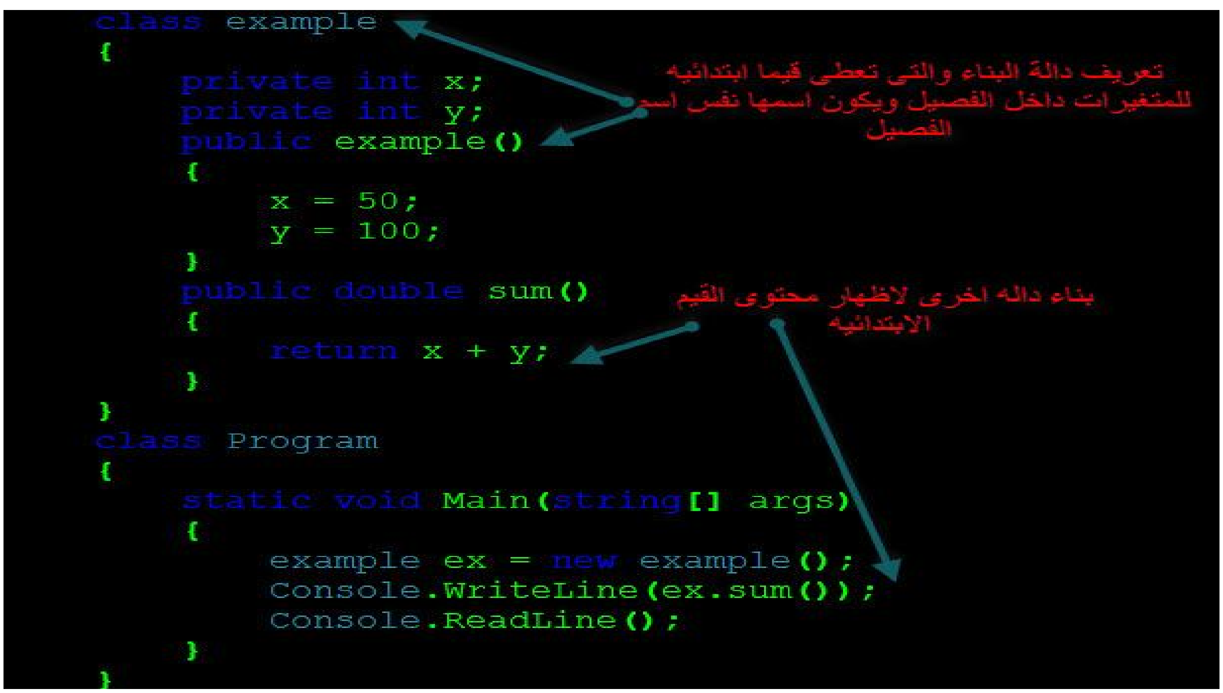
1 – تأخذ نفس اسم Class

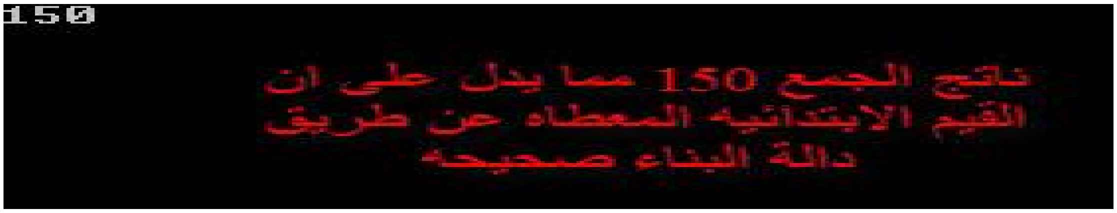
2 – ليس لها قيم مسترجعه

3 – يتم تنفيذ دالة البناء تلقائيا مع كل تعريف لكائن جديد من Class

4 – الكود بداخله يستخدم لوضع Data لاى عضو داخل Class

امثله على ذلك :

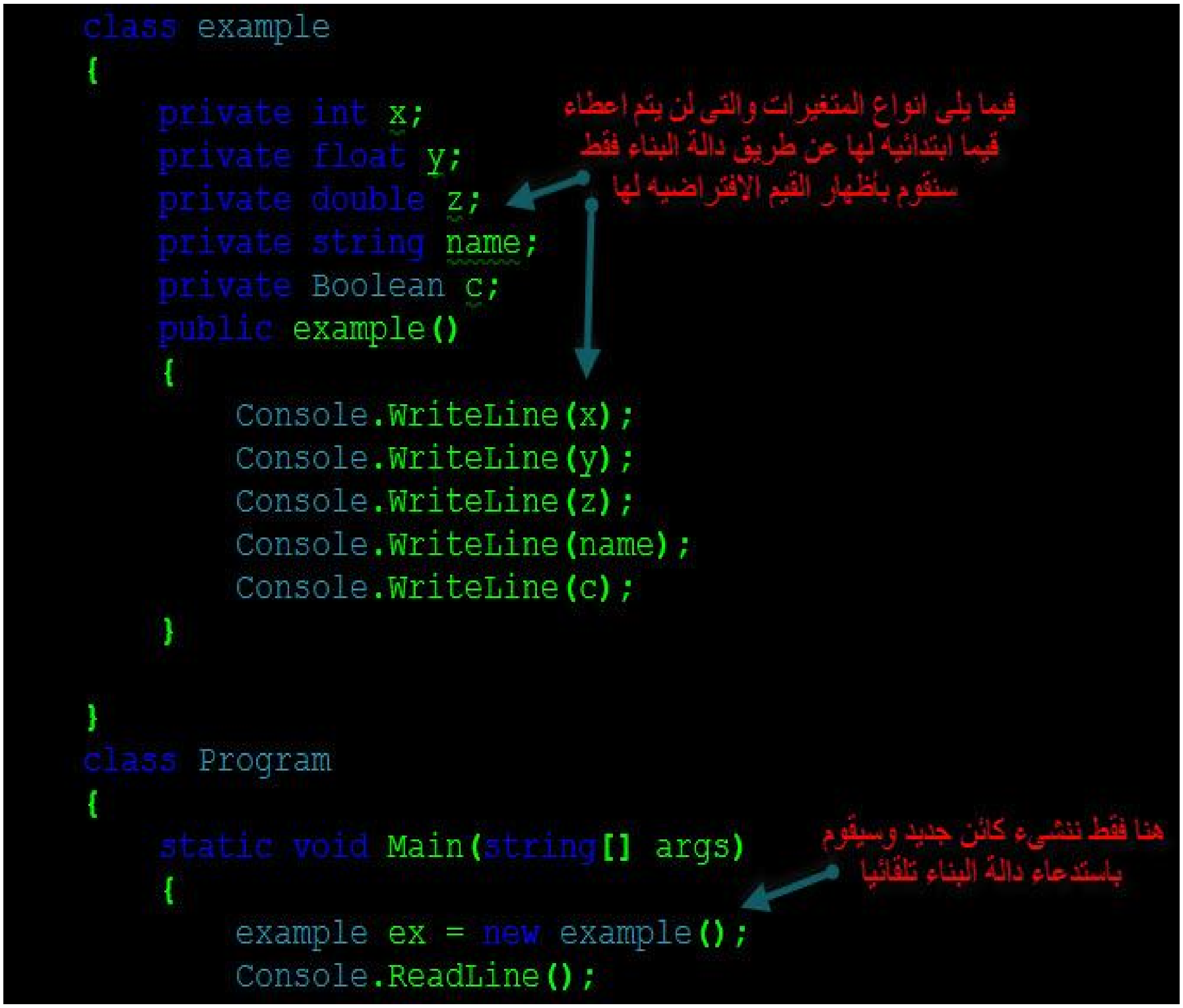


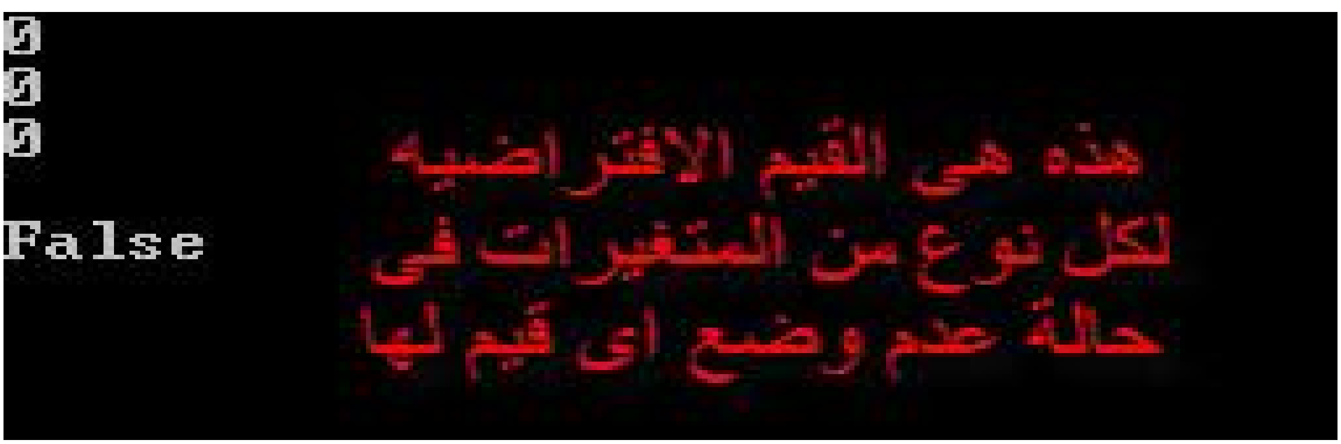


فى حالة عدم بناء هذه الداله بداخل اى فصيل فسيتم وضع قيمه ابتدائيه لاعضائه كالتالى :

String = null
Boolean = false
Int = 0
Float = 0.0

وذلك لان دالة البناء يتم بنائها تلقائيا داخل اى فصيل وهذه ميزه هامه جدا داخل السى شارب حيث انه يتم تخزين هذه القيم داخل الذاكره منعا لتخزين قيم عشوائيه وبالتالى فانت تعرف مسبقا ما الذى يوجد داخل عناوين الذاكره بالنسبه للمتغيرات حتى ولو لم تعطيها قيما ابتدائيه ونستطيع التاكد بالمثال التالى :


class example
{
private int x;
private float y;
private double z;
private string name;
private Boolean c;
public example()
{
Console.WriteLine(x);
Console.WriteLine(y);
Console.WriteLine(z);
Console.WriteLine(name);
Console.WriteLine(c);
}
}
class Program
{
static void Main(string[] args)
{
example ex = new example();
Console.ReadLine();
فيما يلى انواع المتغيرات والتى لن يتم اعطاء
قيما ابتدائيه لها عن طريق دالة البناء فقط
سنقوم بأظهار القيم الافتراضيه لها
هنا فقط ننشيء كائن جديد وسيقوم
باستدعاء دالة البناء تلقائيا


0
0
0
False
هذه هى القيم الافتراضيه
لكل نوع من المتغيرات فى
حالة عدم وضع اى قيم لها

انواع دالة البناء داخل السى شارب :

: Without Parameter – 1

وتستخدم فى وضع قيم ابتدائيه ثابته داخل اى كائن وتعرف بنفس اسم Class وفى هذا النوع يكون لجميع عناصر الفصيل نفس القيم مع تعريف كائن جديد من نفس الفصيل

: With Parameter – 2

وتستخدم لوضع قيم ابتدائيه مختلفه داخل الكائن وفى هذا النوع يكون هناك قيم مختلفه لعناصر الفصيل على حسب البارامترات

```
class example
{ private int num1;
    private int num2;
    //constructor with parameter
    public example(int a,int b)
    {
        num1 = a;
        num2 = b;
    }
   //constructor without parameter
    public example()
    {
        num1 = 50;
        num2 = 100;
    }
}
class Program
{
    static void Main(string[] args)
    {
        example ex1 = new example();
        example ex2 = new example(20, 30);
        Console.ReadLine();
```

دالة البناء بوجود بارامترات لها

دالة البناء فى حالة عدم وجود بارامترات
واستخدام القيم الابتدائيه

الان الداله التى لها احقيه فى التنفيذ تكون على حسب تعريف الكائن

النوع الثانى بدون
بارامترات

النوع الاول مع وجود البارامترات

مثال شامل على استخدام Classes فى البرمجه :

نريد عمل مثال باستخدام Classes وذلك لحساب المساحات لثلاثة اشكال مختلفه

المربع - المستطيل – المثلث على ان يتم الاتى

1 – تعريف المتغيرات اللازمه لايجاد المساحات

2 – بناء داله لاعطاء القيم لهذه المتغيرات

3 – بناء داله لحساب المساحات

```
class square
{int lenght;
    public void setvalue(int a)
    { lenght = a; }
    public int area()
    { return lenght * lenght; }}
class rectangle
{  int wedith, lenght;
    public void setvalue(int a, int b)
    { wedith = a;
    lenght = b;   }
    public double area()
    { return wedith * lenght; } }
class triangle
{ int lenght,tbase;
    public void setvalue(int a,int b)
    {lenght = a;
        tbase = b;   }
    public double area()
    { return (lenght * tbase) / 2; } }
```

```
class Program
{
    static void Main(string[] args)
    {
        square sqr = new square();
        triangle tr = new triangle();
        rectangle rect = new rectangle();
        sqr.setvalue(5);
        tr.setvalue(3, 4);
        rect.setvalue(3, 4);
        Console.WriteLine(sqr.area());
        Console.WriteLine(tr.area());
        Console.WriteLine(rect.area());
        Console.ReadLine();
    }
```

```
25
6
12
```

لكن هناك طرق اكثر احترافيه لكتابة مثل هذه البرامج وذلك لانه تم تكرار نفس المتغيرات ونفس الدوال فى جميع Classes ولتفادى ذلك يجب ان تتعرف على شيء هام جدا يسمى الوراثه داخل البرمجه كائنية التوجه .

Inheritance

ان الوراثه هى عباره عن اشتقاق فصيل من اخر تحمل نفس خصائصها وهذه ميزه هامه جدا داخل البرمجه كائنية التوجه وسبق لنا ان شرحت هذا الجزء فى كتاب هندسة البرمجيات الخاص بى الجزء الثالث منه

يسمى Class الرئيسى بعدة اسماء :

Original class – base class – parent class – super class

ويسمى Class المشتق بعدة اسماء :

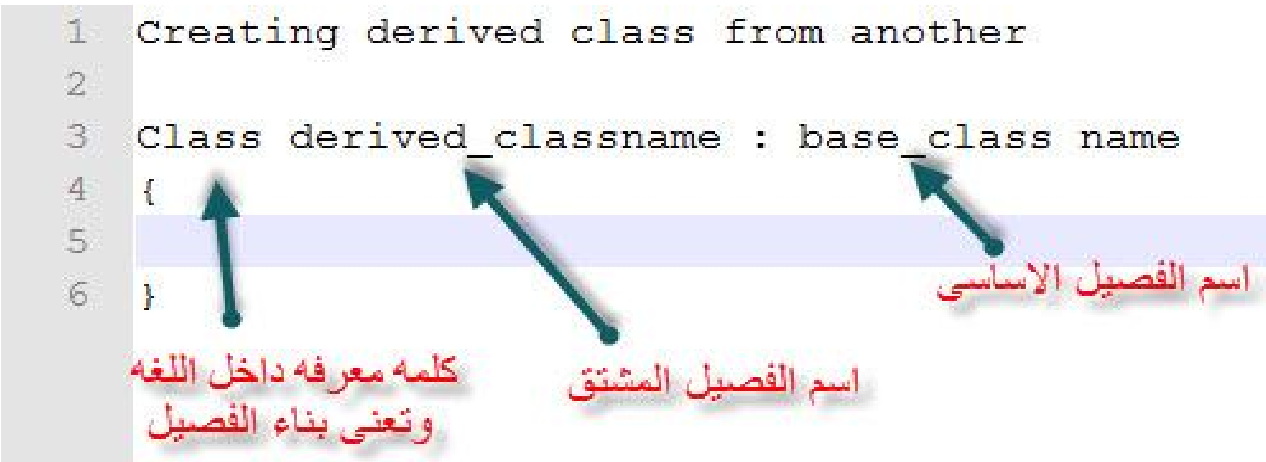
New class – derived class – child class – sub class

ان الوراثه لها ميزتين :

1 – انك تستطيع اشتقاق فصيل من اخر يرث جميع العناصر الموجوده فيه وبالتالى توفر على نفسك عناء اعادة بعض الاكواد يتم تكرارها فى البرنامج كمثال المساحات السابق

2 – انك تستطيع الاضافه داخل الفصيل المشتق الى عناصر الفصيل الرئيسى

القاعده البرمجيه للاشتقاق داخل السى شارب :



يوجد نوعان من الاشتقاق :

1 – النوع الاول انك تشتق فصيل جديد يحمل جميع عناصر الفصيل الرئيسى دون اى اضافات

2 – النوع الثانى انك تستطيع ان تضيف خصائص اخرى للفصيل المشتق

تطبيق الوراثه على مثال المساحات سوف نبنى فصيل رئيسى يسمى الشكل او Polygon ونشتق منه جميع Classes الاخرى




```
class polygon
{protected int w, l;
    public void setvalue(int a, int b)
    { l = a;
      w = b;} }
class squar : polygon
{public int area()
    { return l * w; }}
class rectangle : polygon
{  public double area()
    { return l * w; }  }
class triangle : polygon
{ public double area()
    { return (l * w) / 2; } }
```

لاحظ فى هذا المثال استخدمنا فى تعريف المتغيرات المعرف Protected وليس

Private والسبب لكى تستطيع استخدام المتغير من فصيل مشتق وذكرنا هذا من قبل

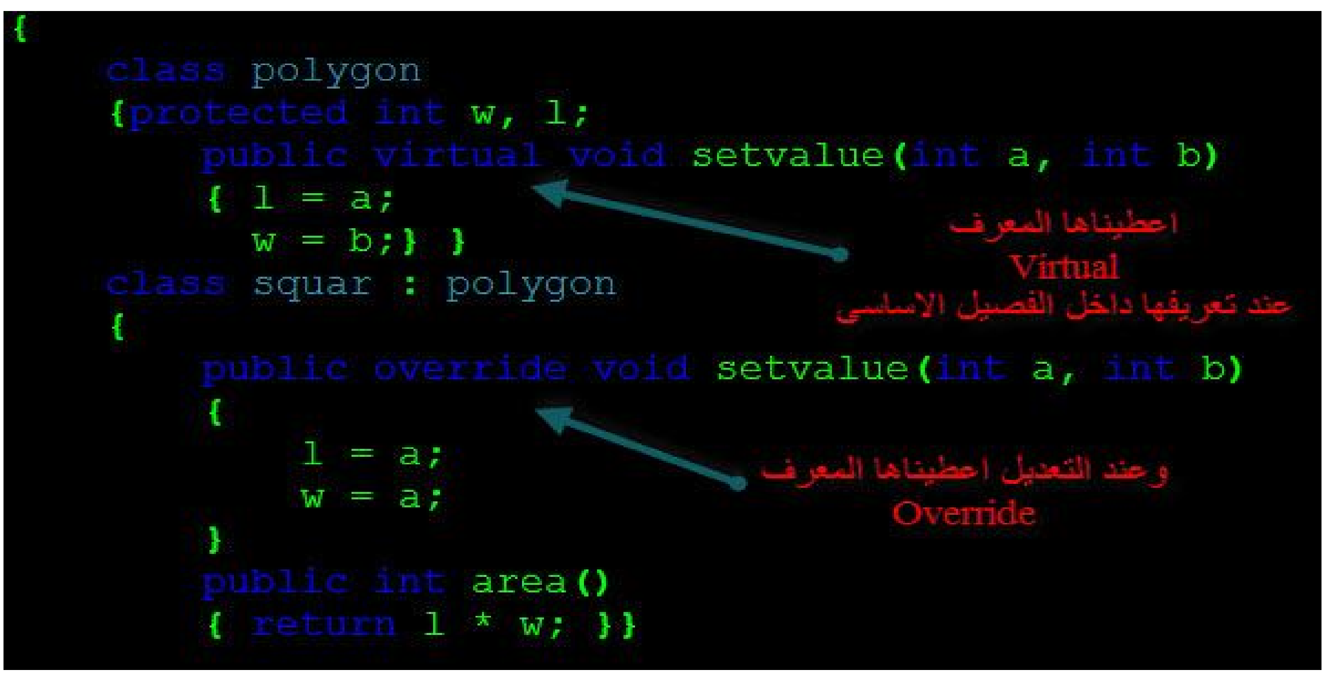
التعديل على بعض الخصائص داخل الفصيل الرئيسى واستخدام قاعدة :

Overriding Base Class Functionality

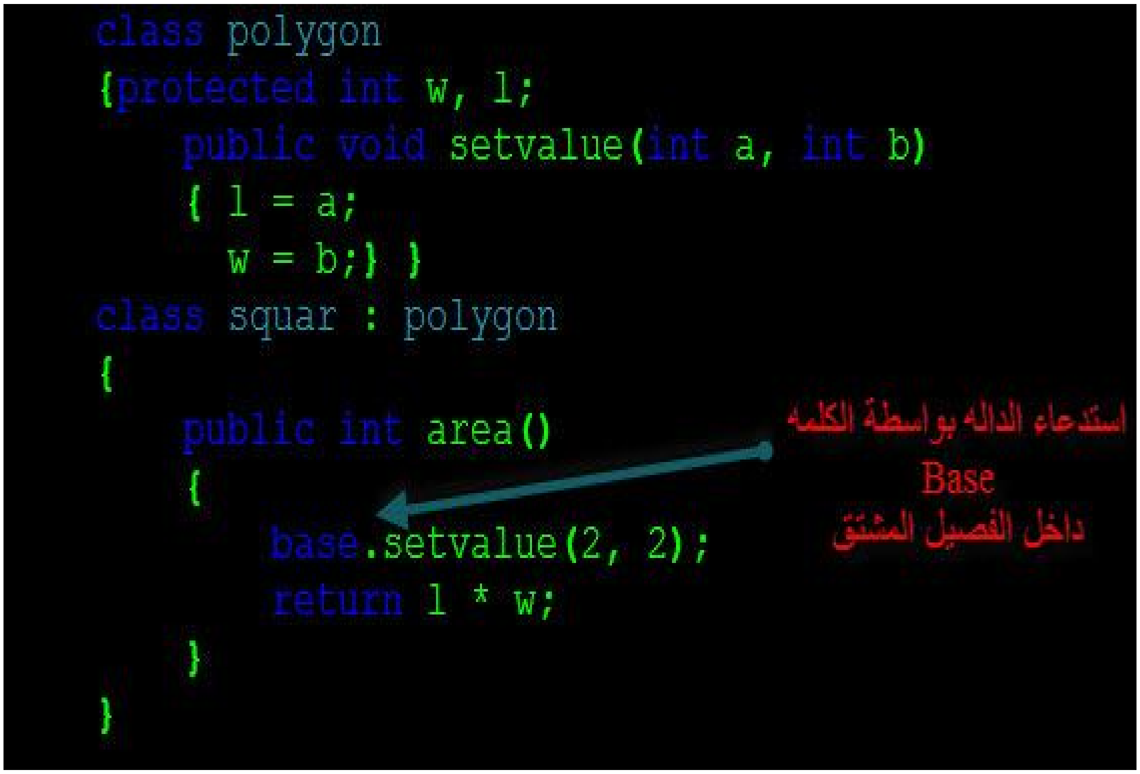
تستطيع التعديل على بعض الخصائص التى كانت موجوده داخل الفصيل الاب وذلك باستخدام الكلمه الثابته داخل السى شارب Overriding واستخدام اسم الداله التى تريد التغيير فيها او ما يسمى Match Signature

وحتى تكتمل قاعدة التغيير وتكون صحيحه فعند بناء الداله فى الفصيل الاساسى والتى تنوى التغيير فيها مستقبليا عند اشتقاق فصائل اخرى من هذا الفصيل فيجب ان تعطى المعرف Virtual

مثال على ذلك فى برنامج المساحات :

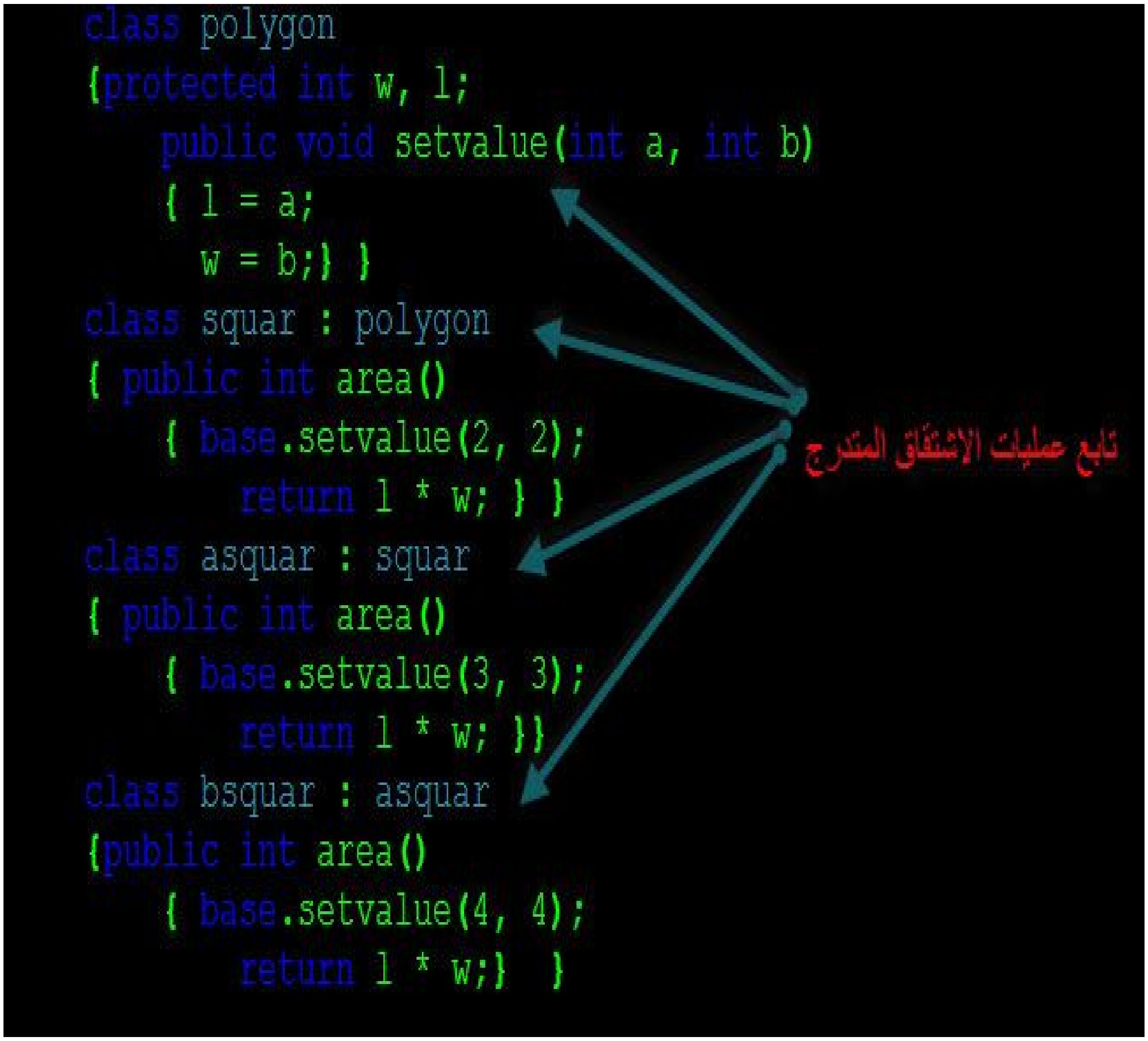


- هناك ميزه اخرى داخل الاشتقاق و هى انك تستطيع استدعاء الدوال فى الفصيل الاساسى داخل الفصيل المشتق وذلك باستخدام الكلمه المعرفه داخل اللغه Base



- : Multi – level Hierarchies

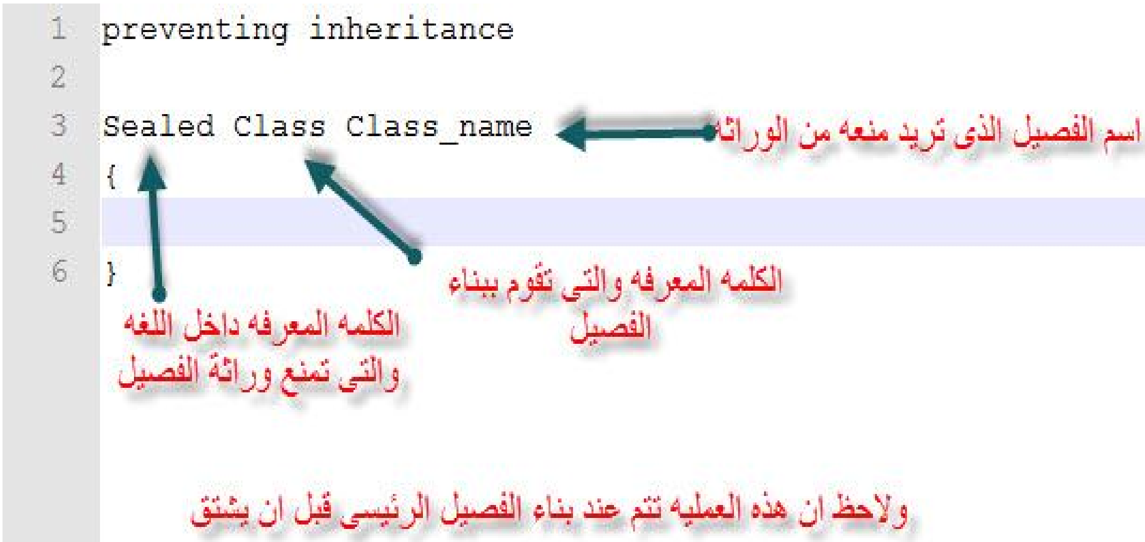
عملية الاشتقاق المتتابع لفصيل من اخر بمعنى ان عملية اشتقاق تمت من فصيل رئيسى تلتها عملية اشتقاق من الفصيل المشتق و هكذا مما يعطيها شكل هرمى ولاحظ انك فى كل مره تشتق فيها فصيل من الاخر تستطيع ان تعدل او تضيف خصائص جديده ويجب ان تلاحظ ان التعديلات لها الاولويه فى التنفيذ عن الخصائص الاساسيه

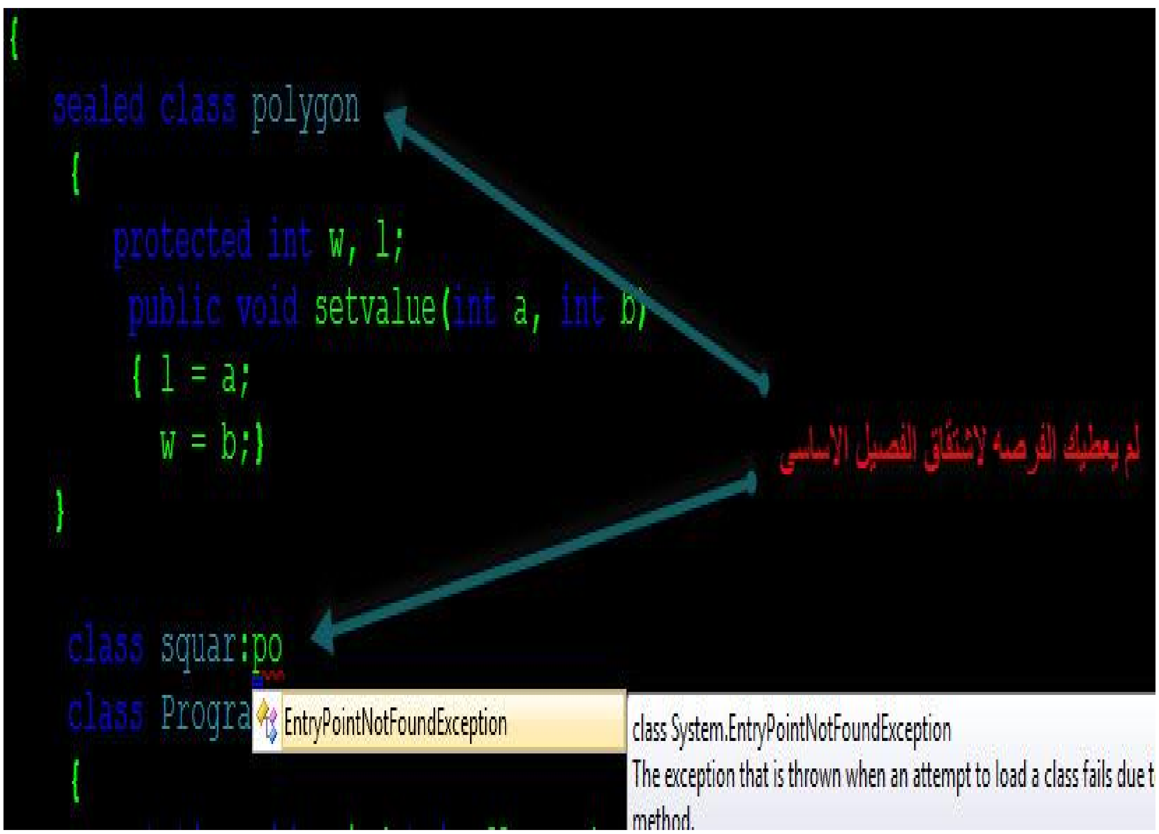


: Preventing Inheritance

منع الوراثه او الاشتقاق داخل البرمجه كائنية التوجه حفاظا على فصيل معين من الاشتقاق وتحدث كثيرا مع Classes المبنيه داخل السى شارب اساسا والتى قامت شركة مايكروسوفت ببرمجتها حفاظا على حقوق ملكية الكود وتسمح باشتقاق بعض الفصائل الاخرى

القاعده البرمجيه لمنع اشتقاق Class :





Polymorphism

ان مفهموم Polymorphism داخل السى شارب هو تماما كمفهوم التعديل على بعض خصائص الفصيل الاساسى وذلك باستخدام المعرف Override ومن الممكن ان تقوم ببناء Class الاساسى دون عمل تكويد له ثم بعد ذلك تعدل وتقوم بعمل الاكواد فى الفصائل المشتقه

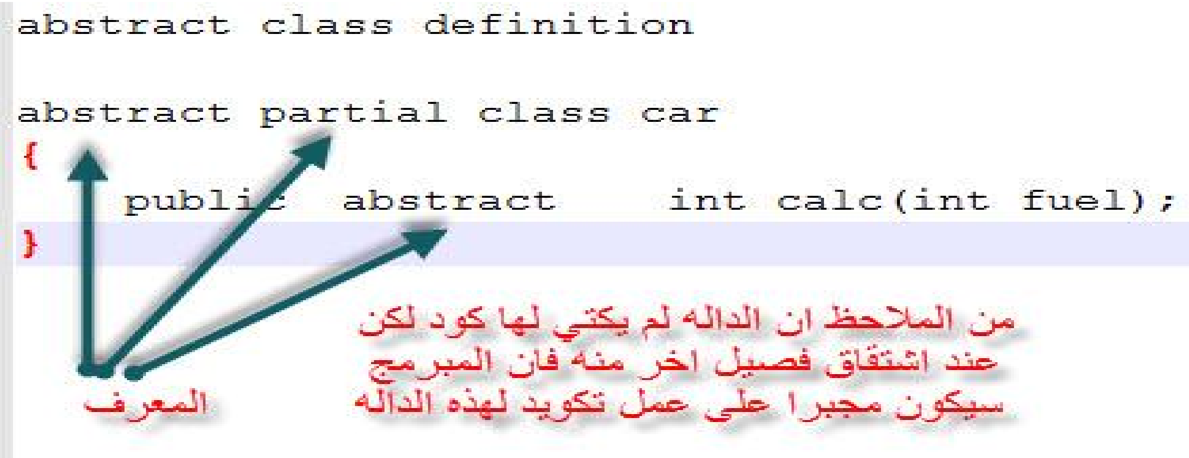
Abstract Class

ومعناه Class بدون كود وهو مجرد Class عادى لكن دون سطر كدو واحد والهدف

1 – تعريف نسخه اصليه من Class على انها Abstract وهى كلمه محجوزه واستعمال كلمة Partial معها داخل السى شارب

2 – جميع الدوال والخواص التى تكتب بداخله يجب ان تعطى نفس المعرف وهو Abstract ولا يجوز تعريف اى متغيرات بداخله او انشاء كائن منه فقط تعريف الدوال والخواص

لدينا المثال التالى :




```
abstract partial class House
{
    public abstract int roomnumber { set; get; }
    public abstract string label { set; get; }
    public abstract double budgetinmonth { set; get; }
    public abstract int roomnumber(int roonmuber);
    public abstract string houselabel(string label);
    public abstract double housebudget(double budget);
}
```

من الملاحظ ان جميع اعضاء الفصيل اخذو نفس المعرف ولا يوجد اى متغيرات
بل خواص ودوال فقط ولا يوجد اى تكويد لهم فقط تعريف لكل منهم

```
class Program
{
    static void Main(string[] args)
    {

    }
}
}
```

لنرى الان اشتقاق class جديد من السابق ونرى ماذا يكون شكله :



```
}
class myhouse : House
{int roomnumber;
    string label;
    double budgetinmonth;
    public override int Roomnumber
    {
        get{return roomnumber; }
        set{roomnumber = value; }
    }
    public override string Label
    {
        get {  return label; }
        set { label = value;   }
    }
}
```

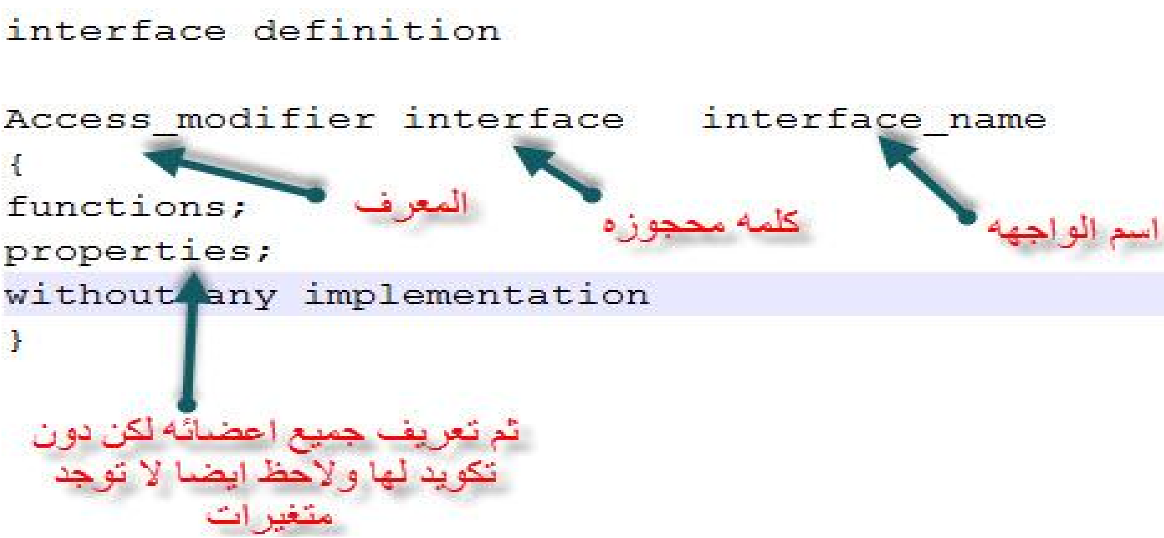
Interface

تصميم الواجهات داخل الكود :

وهى بديل عن تعدد الوراثه الهرميه كما انه يتميز بعمل نسخه منه بالوراثه لكن

الاضافه على الفصائل المشتقه تكون بالتعديل على اعضاء Interface

كما انه يمكن عمل تكويد لاكثر من Interface معا والشكل العام له يكون كالتالى :



ولدينا المثال التالى :



ملحوظه هامه : لا يمكن انشاء كائن من الواجهات او Interface لكن يمكن اشتقاق فصائل منها وبالتالى انشاء الكائنات من classes المشتقه

لا يمكن تعريف المتغيرات داخل interface بدلا منها خواص ودوال والمتغيرات تعرف داخل classes المشتقه

ان عمل interface بديل عن الوراثه الهرميه وتشبه فى عملها abstract class

Types of Errors

1 – syntax error :

هذا هو الاسهل اكتشافه فور كتابة الكود والسبب ان compiler يعترض عليها لانها غير ملائمه للقواعد التى بنيت عليها اللغه او التى تعمل بها داخل لغه معينه

2 – Logical errors :

الاخطاء المنطقيه وهذه الانواع تظهر عند التنفيذ يكون الكود فيها صحيحا لكن سير البرنامج غير منطقى كالقسمه على الصفر مثلا ويسمى هذا النوع Exception او الاستثنائات

3 – Bugs :

هذه هو النوع الاشهر ولايوجد برنامج تقريبا يخلو منه وهى نسيان حذف متغير من الذاكره مثلا عند غلق البرنامج ولذلك تكون هناك عدة نسخ تجريبيه لاصطياد مثل هذه\ الاخطاء

الشكل العام بالنسبه Exception داخل الكود :



Collections

- هى عباره عن مخزن او Container للبيانات ويوفر لك مزايا لا توفرها لك مثيلتها من Array
- ان namespace الذى يحتوى فصيل Array List هو Collections واستدعائه يكون كالتالى :

```
System.Collections;
```

استدعاء فصيل التجمعات

ويعد فئة Array List هى اهم class فيها وتعرف على انها قائمه من Array لها نفس سمات المصفوفات من حيث تخزين مجموعه من البيانات دفعه واحده كالمتغيرات المركبه ولها ايضا مزايا القوائم او List داخل لغة البرمجه فهى اداه اقوى من عمل المصفوفات

طريقة استخدامه داخل السى شارب :

1 – يجب تعريف كائن منه لاستخدام جميع مزايا class

```
create object from ArrayList class:
ArrayList   carlist = new   ArrayList();
```

انشاء كائن من الفصيل ArrayList

```
you can add object :
carlist.Add(temp);
```

تستطيع اضافة كائنات بواسطة دالة الاضافه

```
carlist.Insert(temp);
```

وتستطيع عمل ادراج في جزء معين داخل المصفوفه بواسطة دالة الادراج

```
carlist.Clear();
```

وتستطيع مسح جميع عناصر القائمه دفعه واحده

```
carlist.RemoveAt(4);
carlist.RemoveAt(temp)
```

وتستطيع ازالة عنصر معين برقمه او ازالة الكائن المخزن فيها

```
carlist.Count;
```

وتستطيع معرفة عدد عناصرها باستخدام دالة العد

```
ToArray
```

وتستطيع تحويلها الى مصفوفه عاديه

```
Reverse
```

وتستطيع عكس ترتيب عناصرها

```
Indexof(temp,0);
```

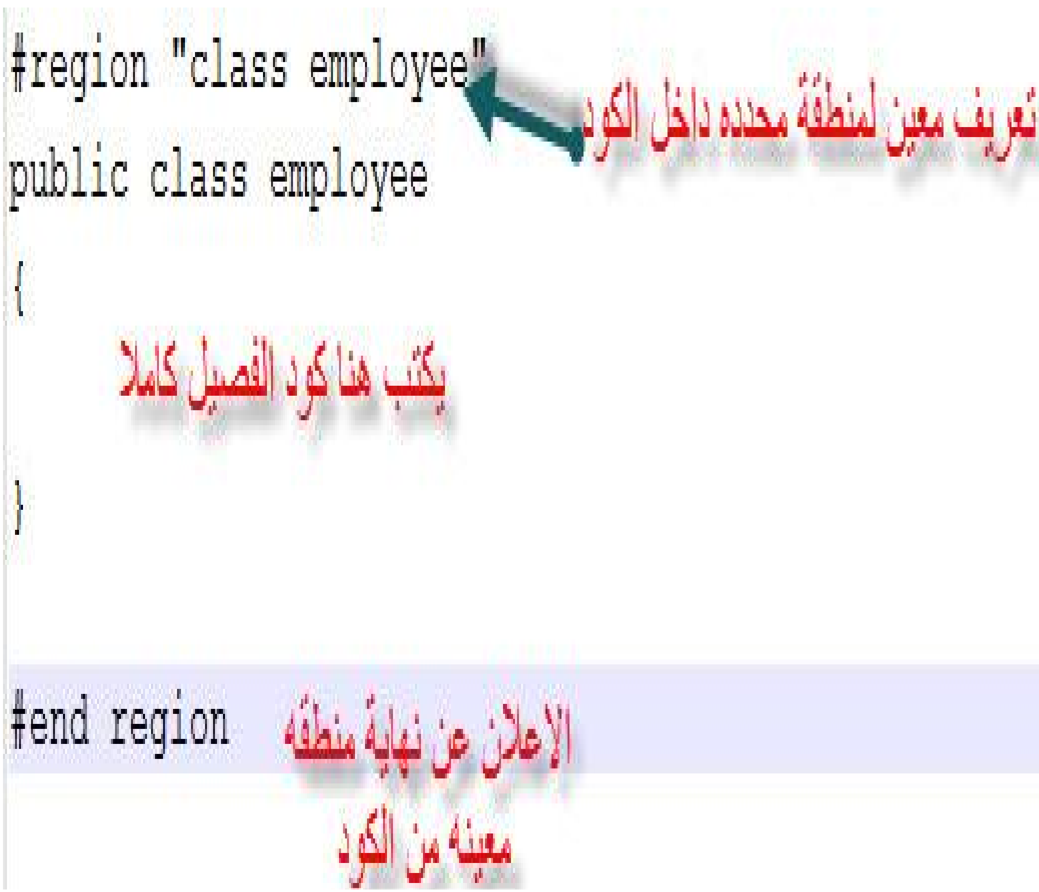
وتستطيع اجراء عملية البحث عن عنصر معين بداخلها

العنصر

بداية البحث

Preprocessor Directives

- وهذه طريقه لترتيب الكود وجعله ضمن نطاقات معينه لعدم التشويش وتستخدم لتحسين وتنسيق مظهر الكود داخل ملف الشفره المصدريه للبرنامج



Comments

- بناء التعليقات داخل الشفره المصدريه :

تعتبر التعليقات جزء هام جدا لا يستهان به خاصه داخل المشاريع التى تحتاج الى شفره مصدريه ضخمه كقواعد البيانات مثلا ولها فائده كبيره جدا عند الرجوع مره اخرى الى الكود وعند عمل الفريق الجماعى

XML Comments:

Important elements in comments :

مقطع بداية التعليق وسيكون بنوع خط مختلف عن الكود

<C> the following must be in differnt fonts

المقطع الذى يحتوى على الشفره المصدريه او الكود

<Code> the following is code

المقطع الذى يحتوى على الاستثنائات او الاخطاء ومعالجتها

<exception> file containg exceptions

المقطع الذى يشرح بارامتر معين داخل الكود ووظيفته

<param> explain parameter in function

عادة لا تستخد جميعها مره واحده فى كل كود تكتبه انما تستخدم حسب الحاجه اليها

///<Summary> المقطع الذى يضم بناء فصيل معين

///employee class of the company

///</Summary>

المقطع الذى يشرح وظيفة القيمه التى تعود من داله معينه داخل الكود

<returns>explain the function result

الشرح المختصر لجزء معين من الكود

<Summary>abstract explain to code

المقطع الذى يشرح الخاصيه والقيمه التى تعطيها لمتغير او

<Value>explain property القيمه التى تعود بها

```
{
   /// <summary>
   /// class of all operations
   /// class operations
   /// <c>
   /// now we build new class contain on all operation
   /// we need
   /// </c>
  ///<code>
   ///
   class Operations
   {int x, y, z;
       public int X { ///<value>thie properity well set value to number x </value>
           set{x = value;}
           get{return x;} }
       public int xsquar(int a) {///<param>a well give the value of x</param>
           x = a;
           ///<return>function well return squar of value x</return>
           return x ^ 2;  }
  ///</code>
   /// </summary>
```

انتهى الجزء الاول من هذا الكتاب من اساسيات البرمجه فى لغة السى شارب

اعتمد على المعلومه المباشره واهتم بها اكثر وعذرا على عدم الاهتمام بالتنسيق

memorycode_84@yahoo.com